Honeywell

# CONCEPTS AND FACILITIES MANUAL
PRELIMINARY EDITION

CP-V

SOFTWARE

# CONTROL PROGRAM-FIVE CONCEPTS AND FACILITIES

July 1, 1976

Ed Bryan
Fran Farrand
Doug Heying

# NOTICE

This manual describes the concepts and facilities of the Control Program-Five operating system. The manual documents the D00 version of the system (with some modifications reflecting E00).

## TABLE OF CONTENTS

APPENDIX

## FIGURES

## TABLES

## GLOSSARY

ANS tape a tape that has labels written in American National Standard (ANS) format.

bandwidth the maximum rate at which memory can deliver or accept information.

batch job a job that is submitted to the batch job stream through the central site card reader, through an on-line terminal (using the Batch processor), or through a remote terminal.

batch job stream a set of jobs which are to be run in the batch mode. These jobs are scheduled by CP-V in a manner that optimizes the use of nonsharable resources.

bipoint line a line that connects a single remote transaction processing station to the computer center. (See multipoint line.)

CAL a CAL1 machine instruction and an associated parameter list which describes the details of a request being made to the system.

CP-V labeled tape a tape that has labels written in a format unique to CP-V.

data control block (DCB) a table in the user's program that contains the information used by the monitor in the performance of an I/O operation.

data set a device which converts data processing device signals to telephone tones and telephone tones to device signals. (Also referred to as "modem".)

data set controller a hardware interface between a remote processing terminal and the central computer.

DCB see data control block.

File Information Table a table of information associated with each file. It controls who may access the file and how it may be accessed.

FIT see File Information Table.

FPT see function parameter table.

function parameter table (FPT) a table through which a user's program communicates with a monitor function (such as an I/O function).

ghost job a job that is neither a batch nor an on-line program. It is initiated and logged on by the monitor, the operator, or another job and consists of a single job step. When the ghost program exits, the ghost terminates.

granule a block of disk sectors large enough to contain 512 words (a page) of stored information.

JCL job control language consisting of control commands. See monitor control commands.

JIT see job information table.

job a unit of work. A batch job is preceded by a JOB control command and consists of all the commands and information which follow that command. An on-line job consists of the entire terminal session. A ghost job consists of a single job step.

job information table (JIT) a table associated with each active job. The table contains accounting, memory mapping, swapping, terminal DCB, and temporary monitor information.

job step a subunit of job processing such as compilation, assembly, loading, or execution. Information from certain commands (JOB, LIMIT, and ASSIGN) and all temporary files created during a job step are carried from one job step to the next but the steps are otherwise independent.

key a data item consisting of 1-31 characters that uniquely identifies a record.

key-in information entered by the operator via a keyboard.

library a collection of frequently-used routines in a form that expedites their inclusion into other programs.

load module (LM) an executable program formed by the Load or LYNX loader, using relocatable object modules (ROMs) and/or load modules (LMs) as input information.

logical device a peripheral device that is represented in a program by an operational label (e.g., BI or PO) rather than by specific physical device name.

logical device stream an information stream that may be used when performing input from or output to a symbiont device. At

SYSGEN, up to 15 logical device streams are defined. Each logical device stream is given a name (e.g., L1, P1, C1), each is assigned to a default physical device, and each is given default attributes. The user may perform I/O through a logical device stream with the default physical device and attributes or he may change the physical device and/or attributes to satisfy the requirements of his job.

MAILBOX file   a file with the name MAILBOX which contains account-specific messages from the system. Each account may have a MAILBOX file associated with it.

modem   see data set.

monitor   a program that supervises the processing, loading, and execution of other programs.

monitor control commands   commands that control the construction and execution of programs and provide communication between a program and its environment.

multipoint line   a line that connects two or more transaction processing stations to the central computer. A line controlled by the computer as though it were connected to two or more stations is considered to be multipoint even though it connects only one station to the computer. (See bipoint line.)

object module   the series of records containing the load information pertaining to a single program or subprogram (i.e., from the beginning to the end). Object modules serve as input to the Load or LYNX processor.

op label   see operational label.

operational label   a symbolic name used to identify a logical system device.

overlay loader   a monitor routine that loads and links elements of overlay programs.

overlay program   a segmented program in which the element (i.e., segment) currently being executed may overlay the core storage area occupied by a previously executed element.

prompt character   a character that is sent to the terminal by an on-line language processor to indicate that the next line of input may be entered.

protective mode   a mode of tape protection in which only ANS expired tapes may be written on through an ANS DCB; no unexpired ANS tape may be written on through a non-ANS DCB; all ANS tapes must be initialized by the Label processor; no tape serial number specification is allowed at the operator's console; specification of an output serial number in an ANS DCB forces processing to be done only on a tape already having that serial number; tapes mounted as IN may not be written; and tapes mounted as other than IN must have a write ring. (See "semiprotective mode".)

public library   a set of library routines declared at SYSGEN to be public (i.e., to be used in common by all concurrent users).

RAD   a fast, auxiliary, fixed-head, random-access disk memory. (RAD stands for Rapid Access Data.)

relocatable object module (ROM)   a program or subprogram in object language generated by a language processor such as Meta-Symbol or FORTRAN.

ROM   see relocatable object module.

scheduler   a monitor routine that controls the initiation and termination of all jobs, job steps, and time slice quanta.

secondary storage   any rapid-access storage medium other than main memory (e.g., RAD storage).

semi-protective mode   a mode of tape protection in which a warning is posted to the operator when an ANS DCB attempts output on a non-ANS tape or an unexpired ANS tape, when a non-ANS DCB attempts output on an unexpired ANS tape, or when a tape mounted as INOUT has no write ring. The operator can authorize the overwriting of the tape or the override of INOUT through a key-in (OVER and READ). ANS tapes may be initialized by the Label processor or may be given labels as the result of an operator key-in; tape serial number specification is allowed at the operator's console; and specification of an output serial number in an ANS DCB forces processing to be done only on a tape already having that serial number unless the operator authorizes an overwrite. (See "protective mode".)

shared processor   a program (e.g., FORTRAN) that is shared by by all concurrent users. Shared processors must be established during SYSGEN or via the DRSP processor.

# CHAPTER 1. INTRODUCTION

## CONTROL PROGRAM-FIVE

Control Program-Five (CP-V) is a comprehensive, multi-use virtual memory operating system. Its name is derived from the following five modes of operation:

- Time-sharing
- Multiprogrammed batch processing
- Remote processing
- Real-time processing
- Transaction processing

CP-V supports these modes of operation with balanced service. There is no inherent emphasis on one mode of processing.

The five modes of operation are also designed so that they may operate concurrently. That is, several programs utilizing different modes may be simultaneously resident in main memory with CP-V selecting the appropriate one for execution at a given time or possibly selecting more than one if the hardware configuration provides more than one CPU for a multiprocessing environment.

Modularity allows the user to select only the mode or modes required by a given task. CP-V performs equally well whether a single mode is used or multiple modes are combined. Common services, file management and procesors facilitate an exceptional degree of compatibility between all the modes of operation. These common services and the fact that they are provided in a rather uniform way to all users regardless of the mode of use has been emphasized in the design of CP-V.

Of particular importance is the file centered nature of CP-V. A comprehensive file system is integrated into CP-V. This means that programs can communicate easily since files are managed in a common way by a common set of programs.

## MODES OF OPERATION

To give the reader a general overview of the system, the five modes of operation will be described briefly in this introduction.

- The time-sharing mode allows more than 128 interactive terminals to be connected to the central computer at one time. Rapid access to and response from CP-V creates an atmosphere in which each time-sharing user appears to have the entire system dedicated to his task.

- The batch processing mode is designed to maximize utilization of the system's resources while preventing conflicts in resource use. Batch jobs may be submitted to the batch job stream through the central site card reader, from an on-line terminal, or from a remote site via the remote processing mode.

- The remote processing mode provides flexible communication between CP-V and a variety of remote terminals. Remote terminals can range from a simple card reader and line printer combination to another large-scale computer system with an assortment of peripheral devices.

- Real-time processing involves reacting to external events (including clock pulses) within microseconds. Selected external events are allowed to interrupt the real-time user's program so that they can be processed at the time they occur. After an interrupt has been processed, control may then return to the interrupted program or may be directed elsewhere.

- Transaction processing facilities provide an environment in which several users at remote terminals may enter business transactions, simultaneously utilizing a common database. The transactions are processed immediately, as they are received, by application programs written especially for the particular installation.

## CP-V FEATURES

The list below outlines those features, facilities or principles which give CP-V a special uniqueness among large-scale, full-function operating systems.

1. Program compatability in all concurrent operating modes. Programs may be run in any mode without modification (subject to very minor restrictions).

2. Single, central file management facility within the system. Files are completely compatibile across operating modes and language processors, eliminating the need for file conversions. The file system is easy to use, provides access security, dynamically allocates secondary storage file space, and is accessed compatibly with devices such as card readers, line printers, tape drives, and user terminals. Files may be updated concurrently by separate programs.

3. Event driven, priority-adjustable scheduler integrated with swapping and virtual memory management.

4. High interactive response at time-sharing terminals which is nearly independent of system load.

5. Multiprocessing with up to four central processing units per system.

6. An excellent remote processing system. It includes dynamic workstation definition, concurrent master and slave operation, and network support. The system supports any HASP-360/20 protocol IRBT as well as IBM 2780, IBM 3780, and DCT-2000 compatibile RBTs. The remote processing system is integrated with symbiont spooling as its buffering mechanism.

7. Ease and naturalness of use for the casual user in both batch and time-sharing modes. Simple yet comprehensive languages exist for control of time-sharing and batch jobs.

8. System default conventions that simplify programming, batch JCL setup, and terminal commands.

9. Time-sharing access to all devices. Devices are symbiont spooled if appropriate.

10. Easy machine language level I/O permitting access to special devices.

11. Time-sharing access to almost all programs.

12. High-power interactive debuggers with combined use of more than one debugger being possible.

13. Terminal personality including typeahead, echoplex, support of virtually any terminal -- all ASCII TTY compatibles, CRT's IBM 2741's, IBM 5100, Memorex, Execuport -- special handling for tabs, paper tape, transparent (uncoverted) I/O, dynamic timing algorithms, all setable for each individual terminal.

14. A terminal may carry the program control stream and/or the program may control many terminals as passive devices.

15. High CPU performance. CP-V has very low overhead for its supplied services.

16. High I/O performance via treed file indexes and several forms of I/O caches and program-disassociated buffering.

17. Efficient use of main memory through a tightly coded and carefully overlayed operating system, as well as efficiently managed user memory.

18. Automatic, operator-free crash recovery with complete preservation of current user file updates, retention of batch jobs awaiting processing or waiting to print, and a complete diagnostic analysis of the memory dump. The seriousness of the recovery is determined and an appropriate level of recovery is automatically chosen.

19. Hardware diagnostics available from time-sharing terminals.

20. Remote access to hardware and software diagnostics.

21. IBM compatibility through ANS and OS labeled tape in either ASCII or EBCDIC and the HASP communication protocol.

22. Excellent protection and security of programs and files.

23. Shared reentrant system processors which may be user-supplied and may be dynamically added or changed during system operation. These include language processors such as APL and BASIC, public libraries such as that for FORTRAN, and user-written programs in FORTRAN and other languages.

24. Command processors for on-line, batch and EASY (GE Mark II) or for installation-specific, specialized processors.

25. A modern, extensive database management system, EDMS, interfaced conveniently to APL, COBOL, FORTRAN, and machine language programs.

26. Ghost jobs for a variety of system and user tasks.

27. A comprehensive accounting system including a dynamic charge rate structure.

28. An integrated performance monitor for measuring and tuning system performance.

29. Simple to use, high-speed system generation process.

30. Completely relocatable and symbolic system patching.

31. A consistent design philosophy yielding an easy-to-modify, modular structure.

32. Small staff requirement for installation and system support.

## ORGANIZATION OF THE MANUAL

The remainder of this manual is organized as follows:

- Chapter 2 provides an overview of the operating system.

- Chapters 3 through 7 discuss elements of the system which are common to all modes of operation.

- Chapters 8 through 12 each describe those features of the system which deal exclusively with the time-sharing, batch processing, remote processing, real-time, and transaction processing modes of operation, respectively.

- Chapters 13 through 16 describe those features of the system which are of interest to the system manager, system programmer, and computer operator.

- Appendix A describes various CP-V processors.

# CHAPTER 2. THE OPERATING SYSTEM

## OVERVIEW

The CP-V operating system consists of a monitor and a number of associated processors. The monitor provides two basic functions:

- Control of the entire system operation, making efficient use of system resources and providing good system response.

- Services to the user that enhance the hardware to provide a virtual machine to each user which is easy to use and is enclosed in an envelope of security.

The associated processors provide specific functions such as compilation, execution, and debugging. All of the processors available for a CP-V system are listed and categorized in Figure 2-1. Some of these processors will be mentioned in the remaining chapters of this manual. A brief description of all of the processors is given in Appendix A.

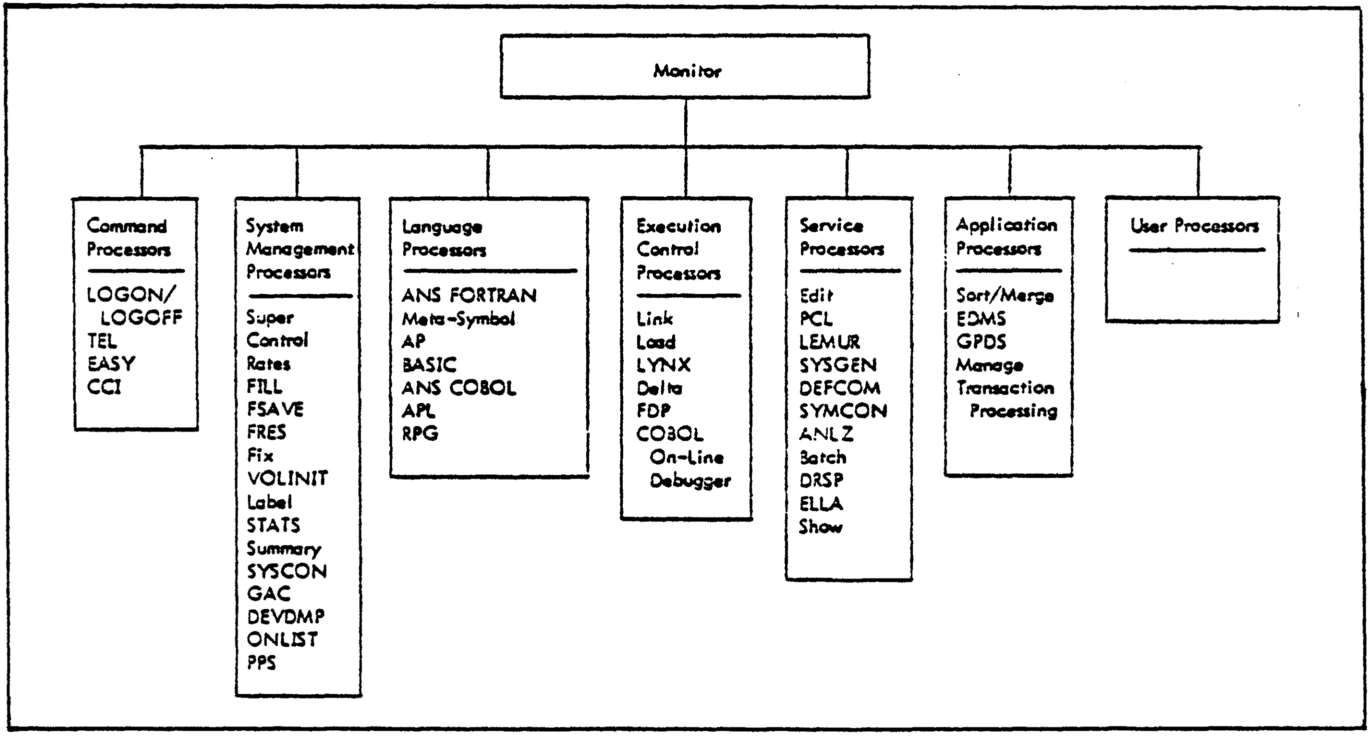


Figure 2-1. CP-V Operating System

Work is performed by the CP-V system through a combination of the CP-V processors and user-developed programs. Each unit of work is packaged together as a "job". There are three different kinds of jobs in CP-V to meet different user (and system) requirements.

- A batch job is one in which the entire control stream and resource requirements (e.g., tapes, spindles) are known to the system before the job is put into execution. Given this information, it is possible for CP-V to schedule batch jobs to optimize the use of non-sharable resources. Generally, batch jobs are disconnected from any human interaction and output is not delivered until completion of the job. Unexpected occurrences in the job will cause the remainder of the job to be eliminated. Batch jobs are submitted to the batch job stream through the central site card reader, through an on-line terminal (using the Batch processor), or through a remote processing terminal.

- An on-line job is one which is connected uniquely to a time-sharing terminal and which receives its control stream from the user at the terminal in an interactive manner. Resource requirements are not made known to CP-V in advance and thus must be acquired on a contention basis. This is workable since a human is in the loop and can make decisions when resources are unavailable. Also, unexpected occurrences need not abort the job since the human is able to correct the condition and continue the job. Other than being unable to pre-allocate resources, an on-line job can do nearly everything a batch job can do.

- A ghost job has no input control stream and is not connected to any terminal. (Many system ghost jobs ask the central site operator for "advice" however.) A ghost job is usually providing some service to the monitor. Its actions are controlled by communication via a file or some other form of internal communication.

## THE MONITOR

The CP-V monitor functions as the major control element in an installation's operating system. In general, the monitor governs the order in which programs are executed and provides common services to all of them (see Figure 2-2).

The number, types and versions of the programs in an operating system vary, depending upon the exact requirements at a

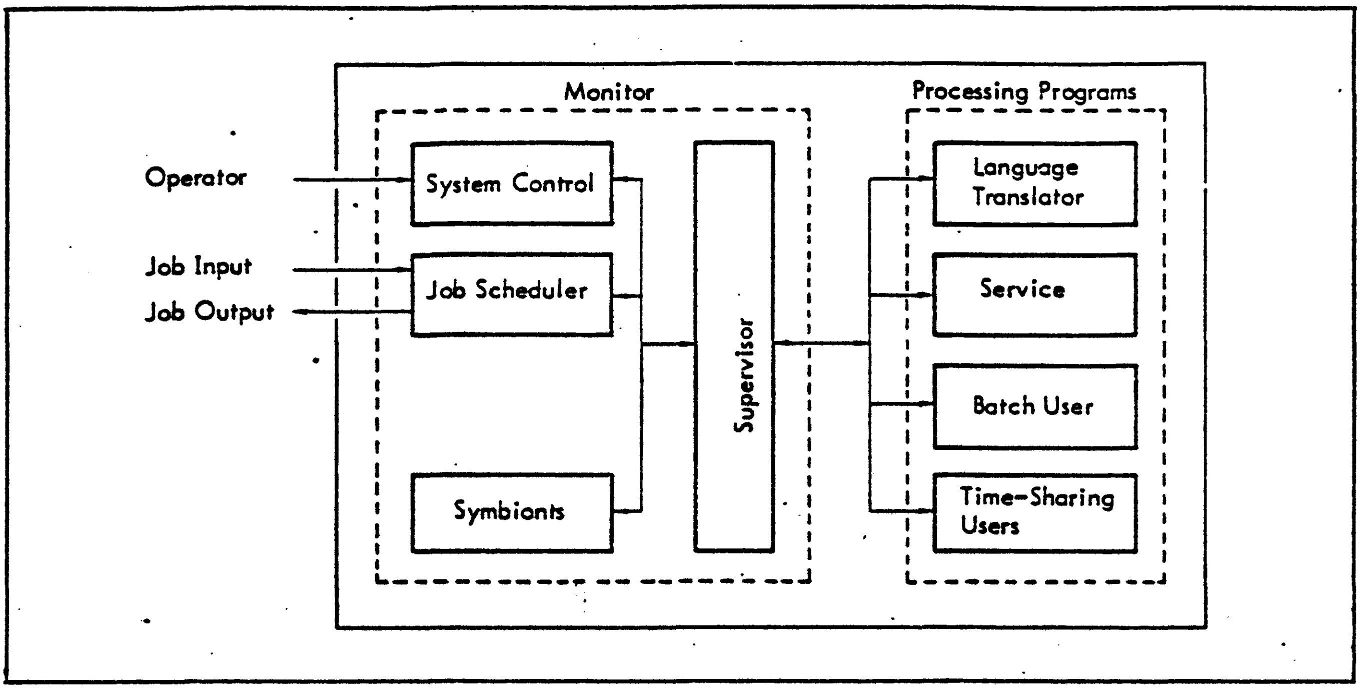


Figure 2-2. Operating System

particular installation. Each operating system consists of a selection of monitor routines and processing programs that are closely integrated for a given range of applications.

The operating system required for a particular installation is defined through use of the System Generation programs. System generation (most frequently referred to as SYSGEN) is performed by the installation's system manager.

As the requirements of an installation increase, the operating system can easily be enlarged, modified, or updated. The ability to adapt conveniently to new requirements is inherent in the system design. Once a system is generated, it can be quickly expanded to include user's programs, data, and system libraries. User's programs and the standard system processors are equivalent in that they are stored, cataloged, and referred to within the system in the same way. They are also written using the same conventions for communicating with the monitor.

The operating system is self-contained and requires operator intervention only under exceptional conditions.

The monitor uses sophisticated techniques for efficient machine operation in a production environment. The ability to process a continuous series of jobs with little or no operator intervention is one of the most important features of the system. By reducing the need for operator participation, the operating system ensures faster throughput, and operations are less subject to error. For the most part, the operator should only have to perform routine tasks such as loading and unloading tape reels.

Complete and easy-to-use I/O services are available to user programs, thus relieving the programmer of many coding chores. Device assignment is general and automatic, enabling the user's program to exploit the complete flexibility of peripheral units.

I/O service is comprehensively organized to simplify programming and make machine utilization efficient. I/O transfers are automatically buffered, and I/O peripherals are serviced on a queue basis (by job). Jobs can thus be executed sequentially even though they might normally be I/O-bound and delay use of the CPU or other I/O devices.

The job scheduler permits selective job operation based on job type or administrative priority to maximize throughput efficiency or environmental needs. The computer operator maintains complete control over the job stack on secondary storage.

Rapid access data (RAD) and disk pack (DP) storage devices are used for secondary storage. Secondary storage management is essential to efficient operation of the monitor, since such storage is fully exploited in various ways. It is used for system storage to overlay portions of the monitor, minimizing core memory residency. Service processors (compilers, assemblers, etc.) are contained on secondary storage for immediate access and they, too, capitalize on rapid overlay techniques to minimize core memory requirements at execution time. Scratch storage for service processors and user programs is available on secondary storage. Finally, the secondary storage accommodates permanent and temporary user files.

User files may be stored on public RAD or disk packs or on private disk packs or magnetic tape. Three file structures are available: random (direct), consecutive, and key-indexed (indexed-sequential). Access may be either direct (keyed) or sequential. Programs coded to access the simpler consecutive files may correctly access the more complex keyed files sequentially without program change. Files are protected from unauthorized use by passwords, by implicit lists of users authorized to execute, to read, or to update them, and by encryption of record keys.

User programs can avail themselves of the secondary storage and the overlay service of the monitor. With these facilities, user programs that require more operating main memory storage than is physically available can be easily segmented and controlled so that only part occupies available main memory at any one time. The monitor accepts the overlay structure of the user's program and ensures proper sequencing and transferring of program elements. It also detects inconsistencies in the logical overlay structure and logs them with a diagnostic message to the user.

The monitor provides for complete accounting of user job activity on the computer. Because of the system's multiusage capability, the accounting information indicates both elapsed time and actual machine facility utilization of each job.

The monitor provides job accounting and validation of each user's job activity:

- Validity or authorization checks are made on the user's name and account number combination. Jobs are aborted when the name and account number are not previously authorized by the installation manager.

- A discrete accounting record is written at the termination of each batch job.

- Standard accounting can be supplemented by the user supplying initiation and termination routines for a job.

The monitor's memory management function relocates user programs into the currently available core memory space, satisfies all library subroutine references, and links all program elements called for by the user. In addition, run-time debugging calls are recognized and established for the programs to be run.

The monitor responds to the moment-by-moment requirements of controlling machine operation, switching between programs requiring service, and providing services at the explicit request of the user's program. The monitor processes that perform these functions are listed below:

1. Basic Control.

2. Scheduling and Swapping.

3. Multiple Processor Scheduling.

4. Memory Management.

5. File Management

6. Multibatch Job Scheduling.

7. Job Step Control.

8. Terminal I/O Handling.

9. Symbionts and Cooperatives.

10. System Integrity.

11. Initialization and Start-Up.

12. Operator Communications.

13. Batch Debugging.

14. Load-and-Link.

15. System Debugging.

The basic control system is an I/O interrupt service and handling routine. It includes trap and interrupt handlers, routines that place requests for I/O in a queue, and basic device I/O handling routines.

The scheduling and swapping modules make the decision to swap, select the users to swap in and out, set up the I/O command chains for swap transfers, and select the next user(s) for execution. They also ensure that any associated, but not currently resident, shared processors are brought in with each user. Special algorithms control I/O scheduling and the balance of machine use between on-line and batch.

Multiple processor scheduling schedules tasks for idle secondary processors after scheduling any possible swaps and prior to scheduling a user for the primary processor. Since secondary processors operate at their greatest efficiency when assigned users with little immediate need for monitor service, a short-term history of service requests is kept for each user. This service rate history is compared against a threshold value set by the installation manager so that only the most promising candidates for secondary processors are selected.

The memory management module controls the use of core and disk storage. Specifically, it controls the allocation of physical core memory, maintains the map and access images of each user, services the "get" and "free" service calls for memory pages, and manages the swapping disk space.

File management routines control the content and access to physical files of information. These routines perform such functions as indexing, blocking and deblocking, managing of pools of granules on RADs and disk packs, labeling, label checking and positioning of magnetic tape, formatting for printer and card equipment, and controlling access to and simultaneous use of a hierarchy of files.

The multi-batch job scheduling routines select jobs to be run from the waiting input queue depending on priority, position in queue, and resources available within partitions defined by the installation.

Job step control routines are entered between major segments of a job or an on-line session. They perform the monitor functions required between job steps such as

1. Processing error, exit, and abort CALs.

2. Handling monitor aborts.

3. Processing interpretive exits to associate shared processors or to load program modules.

4. Merging DCB assignments for execution.

Terminal I/O handling routines perform read-write buffering and external interrupt handling for I/O directed to user terminals. These routines also translate character codes, insert page headers and vertical format control characters, simulate tabs, and perform other formatting tasks.

Symbiont routines transfer data from the card reader to disk storage and from disk storage to the card punch or line printer. Input cooperatives intercept card read commands in user programs and transfer data from disk storage where it has been stored by the symbiont routines. Output cooperative routines intercept output directed from a user program to a line printer or card punch and transfer the data to disk storage.

System integrity facilities provide error detection and recovery capabilities. This includes security to user files and automatic high-speed restart in case of system failure. Sufficient information is recorded to isolate errors and failures caused by hardware or software.

Initialization and start-up routines are stored on tape and are booted into core storage. After they are in core, they load the monitor root into core and turn control over to the root. The monitor root then completes the initialization of the monitor by starting and running the program called GHOST1 which completes the patching of the system and the initialization of the swapping disk and hardware.

Operator communication routines provide for communication between the monitor and the operator. They transmit messages to the operator and process key-ins received from the operator.

Batch debugging routines provide batch programs with debugging capability through the use of procedure calls. Any batch program may take a snapshot dump of a specified segment of memory, either on an unconditional or a conditional basis.

Load-and-link routines give batch programs two types of loading and linking capability. Through the use of procedure calls, a batch program may:

1. Store the calling program on disk storage, load the called program into main memory, and transfer control to the called program.

2. Load a program into main memory, transfer control to the called program, and release the core area used by the calling program.

System debugging routines provide debugging services to system programmers.

# CHAPTER 3. SCHEDULING, SWAPPING, AND RELATED TOPICS

This chapter contains a discussion of several key CP-V features. They are

- Scheduling and swapping
- Resource management
- Privilege, protection, and security
- Shared libraries
- Shared processor facilities

## SCHEDULING AND SWAPPING

The most vital part of a multi-use operating system is the scheduler, the module whose primary responsibility is allocating the computer's resources. The scheduler is critical in providing fast response to on-line users and rapid throughput for all jobs. The degree of efficiency with which the scheduler performs is the key to optimum utilization of a computer system - and the value of the computer to an organization.

The CP-V scheduler performs two major functions in achieving the goal of high performance:

- Selecting the highest priority user whose program and data are in memory for execution.
- Ensuring that remaining high priority users are in memory ready to use the processing resource when it becomes available.

The CP-V scheduler accomplishes this by

- Creating prioritized status queues into which every job has a single entry.
- Assigning every job in the system a priority.
- Modifying a job priority as requirements for access to the processing resource change during execution.

There are three fundamental classes into which the various status queues may be segmented:

- Waiting to Execute - This group of queues contains those jobs requiring the processing resource in order to proceed.
- Executing - This queue consists of a single entry for each central processor: the job currently using the processing resource.
- Non-Executable - This group of queues contains jobs waiting for an "event" to occur before requiring access to the processing resource.

A primary benefit of the priority queue structure is that it provides complete service to I/O users while concurrently keeping the processing resource busy with compute-bound jobs, allowing maximum utilization of both I/O devices and the basic processor.

Each job is assigned a base priority when first activated. The base priority may be different depending upon the selected mode of operation - for example, batch or on-line - thereby allowing one class of jobs to gain preferential service. During normal operation, the priority of a job changes frequently during processing. Conditions or events that cause the scheduler to modify a job's priority include

- Completion of an I/O operation
- Opening or closing of a file
- Completion of a time quantum
- Addition of real-time task
- Completion of terminal input

Upon being notified of the occurrence of one of these events, the scheduler changes the priority of the associated job. To facilitate the changing of job priorities, the scheduler uses an event-transition technique. This technique can be viewed as a matrix where one coordinate represents all possible events that can occur and the second represents the status queues. Any intersection defined by the occurring event and the current state of the associated job determines the new priority and the new queue. Because the executing programs and the environment continually apprise the scheduler of their requirements and resources available, the scheduler can efficiently and effectively optimize the entire system. Dynamic system tuning is a major factor in making CP-V the most efficient multi-use operating system available.

Another mechanism used by the CP-V scheduler to increase the amount of time spent in processing user jobs is the use of bounded time intervals. QUAN, QMIN, and SQUAN are three time intervals which may be set to ensure that no user job receives more than its share of the processing and memory resources, yet still gets enough to satisfy the users' requirements.

- o QUAN is the maximum time-slice allowed a compute-bound user before another job is given control of the system. This assures that no single compute-bound job of a given priority can dominate the processor resource. The QUAN value is separately specified for each batch partition and all on-line users.

- o QMIN is the amount of processor time guaranteed a job unless pre-empted by a <u>critical</u> real-time task. The processor will still respond to I/O interrupts and perform other system tasks, but the processor will not be given to another user until the current user has received the QMIN time.

- o SQUAN is the minimum memory-resident processor time a job is guaranteed once it has been brought into memory. The job will not be considered for out-swap until it has received the SQUAN time.

The swap scheduler ensures that the highest priority jobs currently requiring the execution resource are in memory. It does this by moving other jobs to secondary storage and moving the highest priority jobs into memory. Candidates for jobs to be moved to secondary storage are located in the same prioritized queues used by the task scheduler. When at the high end, they have a high priority for swapping into memory and for execution; when at the low end, they are prime candidates for removal to secondary storage. This latter feature - that of having a priority for removal of users to secondary storage - is an important aid to efficient swap management. It avoids extraneous swaps by making an intelligent choice about outgoing as well as incoming users.

Another way of viewing the swap storage is as a "virtual" extension of real memory. This virtual memory system allows for total user memory space well in excess of 15 million bytes to be satisfied by a real memory system as small as 256K bytes. And it does this while maximizing the user productivity of the vital execution resource. This results in extremely fast response to on-line users and fast throughput for all jobs.

The CP-V swapper is efficient because it takes advantage of other CP-V modules as well as the capabilities of the hardware on which it runs.

When a user requests service from the system, space is allocated to him on a secondary storage device. As his memory requirements grow during his session, additional area necessary to contain him grows into adjacent sectors following his initial allocation, so that the individual user's space is organized for swapping as quickly as possible.

When multiple users are being swapped out the I/O commands for each user are sorted and chained together, so that the end of one user's area is the shortest possible distance from the beginning of the next. After this chaining is accomplished, the current position of the swapping device is sensed and the swap is begun with the user closest to the current position. This procedure effectively reduces latency well below the average access time for the secondary storage device. Similar logic is applied to the swapping in of multiple processors along with user programs.

The multi-unit, multi-ported memory of the computers on which CP-V runs allows true swap/compute overlap to be achieved: while one user's program utilizes the computing resource, other programs migrate between primary and secondary storage. Because these operations can occur simultaneously, the execution processor spends a maximum amount of time performing productive work. The execution processor does not have to remain idle while a swap is in progress, and swaps can proceed without interfering with computation.

Other swap efficiencies are gained through the reduction of required swaps. Most of the CP-V processors are reentrant, which means the same copy can be used by multiple users - eliminating a requirement for additional primary memory which would otherwise be necessary. Reentrant processors have another advantage: because they are pure procedure, they do not have to be swapped out, since an identical copy already exists on the secondary storage device.

## MULTIPROCESSING

The basis for a CP-V Multi-Processing system is the large memory mono-processor system with one or more additional processors able to access all of memory. This ability for all processors to access all of physical memory is fundamental since the majority of communications between processors is through memory. Thus, all processors are tightly coupled via shared main memory.

Only one of the processors, called the primary processor, executes the entire body of code comprising the CP-V monitor; all other processors, called secondary processors, operate as compute peripherals for use by the primary processor. The secondary processors are assigned slave mode, compute tasks as user units by the primary processor. Once a task begins execution on a secondary processor, it continues until either a quantum-end condition occurs or a need arises to perform a monitor service requiring interaction with stored central tables. When a secondary processor is no longer able to continue execution of its assigned user, the environment is saved and the primary processor is signalled via flags and data in main memory. An interrupt is directed to the primary processor by the secondary only to hasten the detection of these flags by the primary. Upon noticing the idle condition of the secondary, the primary parks the user previously assigned and attempts to find a new user for the secondary. In the event that no suitable user is found, additional attempts are made to schedule tasks for each idle secondary procesor after scheduling any possible swaps and prior to scheduling a user for the primary.

## RESOURCE MANAGEMENT

The term <u>resource</u> has a very specific meaning in the following discussion. A resource is any portion of the CP-V installation that is to be shared by the users in a manner such that each user requiring the resource is allocated the resource for its exclusive use. (An exception to this is private disk packs which under some circumstances may be shared even though they have been defined to be resources.) Peripheral devices and core are common types of resources. Symbiont devices and public storage devices can never be defined to be resources because they are non-allocatable devices; that is, they are <u>never</u> reserved for the exclusive use of one user.

There are special resource management routines within the monitor. The specific task of these routines is to keep track of the number of resources of each kind in use and the number of resources of each kind that are available for use. For a batch job, the requirement for resources is compared with the available resources and the job is not started unless sufficient resources are available. (The user specifies his resource requirements on the LIMIT control command.) Further, the resources are reserved for the exclusive use of the job so that it is guaranteed that they will be available even if a long time elapses between job startup and actual use of the resources.

CP-V does not require that an actual physical device correspond to each of the resources it manages. When there is no correspondence between a resource and an actual physical devices, the resource is called a pseudo-resource. Pseudo-resources are often used to achieve special job scheduling tasks requiring access, for example, to a particular execution partition.

The system manager must define what the resources are for the installation, establish system defaults and maximums for use of the resources, and set limits on the use of the resources for the individual users. He performs these tasks using the following processors:

- SYSGEN PASS2
- Control
- Super

In the PASS2 phase of System Generation, the system manager uses the :RES command to establish which portions of the installation are to be resources. For each resource, he establishes the amount of the resource that may be allocated to all concurrent batch jobs, to all concurrent on-line jobs, and to all concurrent ghost jobs. He also establishes the default amount that is to be allocated to each batch, on-line, and ghost job in cases where the amount is not otherwise specified and the maximum amount that may be allocated to individual batch, on-line, and ghost jobs.

The following types of resources are always defined at SYSGEN - if not explicitly, then by default:

CO - core (main memory)

MC - maintenance console

A maximum of 13 more resources may also be defined.

The Control processor can be used to dynamically modify the default and maximum values associated with each resource. Resources must be defined at SYSGEN. New resources cannot be added to the system via the Control processor. However, a resource may be effectively removed from the system by appropriate modification of the values associated with the resource.

The <u>Super</u> processor is used to establish the maximum amount of each resource that is to be available to each user when the user runs in the batch or on-line modes. In special cases, an individual may be authorized a resource maximum which is higher than the system maximum to allow a special job to run when no other user can acquire that amount of resource. For example, the maximum for core could be set low during the day for pushing through a lot of small jobs, but an individual critical job could be run with a high core requirement.

In order to coordinate the sharing of a CP-V installation among many users, it is necessary to impose limitations on the execution of user programs. These limitations fall into two categories:

1. <u>Service limits</u> which limit such things as:

   - o Job execution time.
   - o Pages of printer output.
   - o Number of cards punched.
   - o Amount of temporary public storage.
   - o Amount of permanent public storage.

2. <u>Resource limits</u> which limit the number of resources of each type that are available for the job (e.g., tape drives, spindles, etc.).

Limits are established, changed, and collected from four sources:

1. <u>SYSGEN PASS2</u> processor - for establishment of system limit tables which define limits to be associated with each batch, on-line, and ghost job. These limits are established through use of the :RES, :BLIMIT, :OLIMIT, and :GLIMIT commands.

2. <u>Control</u> processor - for dynamic modification of the system limit tables.

3. <u>Super</u> processor - for establishment and dynamic modification of the limits for each individual user. The limits are recorded in the :USERS file, a file which contains one record for each authorized user at the installation.

4. <u>LIMIT</u> control command - for establishment of limits on

a particular instance of execution. (The LIMIT control command is only applicable to the batch mode.)

The sequence by which the ultimate service and resource limits are placed on an executing user program is depicted in Figure 3-1. When the job is started, limit values for the job are initially set from the :USERS file record. Values which are not given in that record are then set from the monitor limit tables. For batch jobs, limit values are set to the value specified by the LIMIT control command.

Finally, these composite values are compared to the maximum values in the :USERS table or monitor limit tables and the job is aborted if the limits are exceeded.

The process may be divided into two cases: first, when there is no user maximum specified in the :USERS file record for the limit in question, and second, when there is a user maximum specified. The algorithm applies both to service limits and to resource limits identically, except when noted.

Case 1: No User Maximum in :USERS file

The limit is set to the limit on the LIMIT control command if any. Otherwise, it is set to the system default. If the limit is less than or equal to the system maximum, the job is run. Otherwise, the job is aborted.

Case 2: User Maximum specified in :USERS file

If no LIMIT control command is included with the job, the limit is set to the user maximum for all service limits and all on-line resource limits. The limit is set to the user maximum or the system default (whichever is smaller) for batch resource limits and for job execution time.

If a LIMIT control command is included with the job, the limit is set to the limit on the command if it is less than or equal to the maximum specified in the :USERS file. Otherwise, the job is aborted.

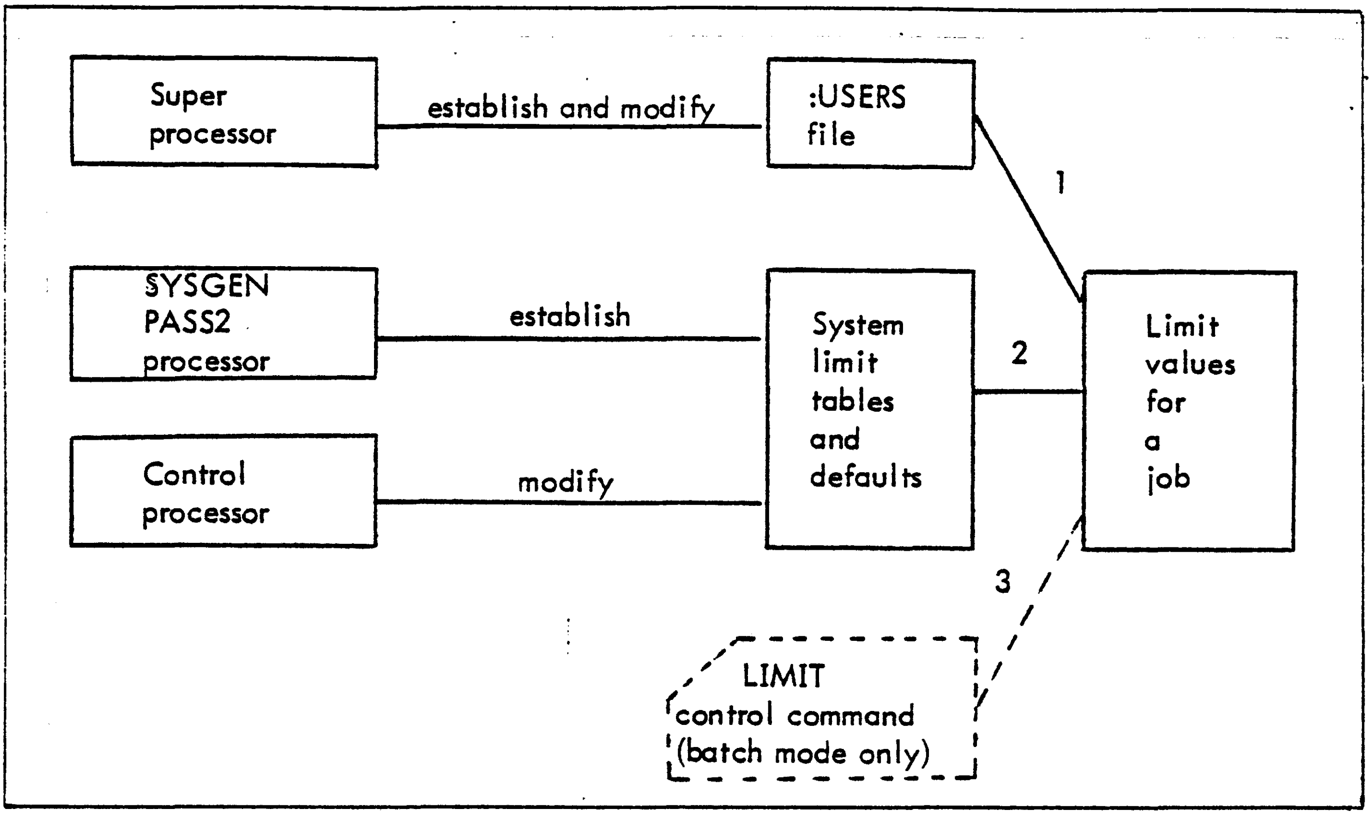


Figure 3-1. Establishing Limits for a Job

## PRIVILEGE, PROTECTION, AND SECURITY

Aside from reliability, one of the major advantages of the field proven nature of CP-V is that it has evolved into a system that stresses total security and preservation of user files with full protection of the operating environment from malicious or inadvertent destruction. Hardware security is provided through the memory map, the map's associated memory access protection scheme, and the memory write protection system of locks and keys. These are discussed below.

1. <u>Memory Map</u>

   User memory is logically divided into 256 pages of 512 words each, permitting any one user at any given time to have a virtual memory of up to 128K words. While many users of the same virtual addressing may be resident at the same time, scattered over the real memory available, the memory mapping system prevents any one user from getting at the physical memory of other users.

2. Access Protection

Memory access protection assures that a user or slave program writes only to its own data area. No access is granted to write over either its own executable code or over any part of the monitor. A two bit access code is associated with each page of a user's virtual memory. The four possible combinations of the code mean:

11 - The program may not access this page.

10 - The user may read but neither execute nor write over this page.

01 - The user may read or execute the data in the page.

00 - The user may read, write or execute the data.

The hardware automatically prohibits any user from violating this code, but allows the operating system and its related routines to bypass the access protection. The result is that the feature prevents a user program from damaging itself, a shared library, or its context.

3. Memory Write Protection

This feature provides individual memory write protection capabilities on a page basis for memory through a system of memory keys and locks. It is used to protect monitor procedure from itself as well as from any I/O operations.

User authorization security is achieved in the following way. Before any user can do any processing on CP-V, an account must be created for him by the system manager. When the account is created, the system manager must specify an account name and an account number. In addition to these items, the following additional parameters may be supplied for each account:

1. The associated type and level of privilege granted the user. The user may be allowed to

   o Utilize real-time services.

- o Bypass security and account checks.
- o Access and change the monitor.
- o Read and write error file; request the devices; invoke diagnostics; authorize enqueue/dequeue automatically.
- o Examine (but not change) the monitor.

2. The password that is to be associated with the account. If specified, no one can do any processing unless the password is provided.

3. Whether or not all files created under this account may be read by other users. This is a general restriction which, if not applied, can be invoked on an individual file and/or user basis through another feature of CP-V.

4. Whether or not a security check is to be performed on newly allocated core and disk storage to be used by this account. If requested, all core and disk memory that the user will access will be effectively erased before being accessed.

5. Whether or not the processors available to this account should be restricted.

6. Whether or not to automatically connect a user of this account to a given program.

Through these features an installation has numerous security controls over each and every user. These controls may, with the system manager's discretion, be applied to users on an individual account basis.

With most systems, the integrity and security of the file system is critical to ensure safe and reliable operation.

The CP-V file system uses three separate techniques to ensure its reliability and to prevent unauthorized access.

1. Password

   A user may assign a private password to a file. If someone tries to access that file without using the proper password, access is denied.

2. Access Lists

   Each file has three lists of account numbers associated with it. The first list says what other accounts may execute that file. It also identifies what processor may be associated with the file. A primary use for this feature, of course, is to control the reading of proprietary programs. If a program is stored as an execute-only file, users can execute the code but never examine it. A second list specifies what users can read or execute the file. They cannot write to it, however. This allows data to be shared without being destroyed. The third list allows users to read, write, or execute the file. This is the list of all users who have complete access to the data or program. Instead of specifying a series of accounts, the creator of the file can say ALL or NONE in any list, which means everyone or no one (but the creator) can access it. Low priority users are restricted from reading all files cataloged under an account other than their log-on account unless:

   a. The log-on account is specifically mentioned in the read access list.

   b. The read access list consists of the single word PUBL.

   A significant option to this system consists of limiting access to a given file only via a specific processor. Thus users in other accounts may be permitted access to such files but only under very special circumstances and only through a system processor.

3. Data Encryption

   Each record of a KEYED or CONSECUTIVE file can be encrypted with a unique encryption key.

In addition to the above (which applies to disk storage), the following are two modes of protection for ANS labeled tapes:

1. Protective Mode: A mode of tape protection in which only ANS expired tapes may be written on through an ANS data control block; no unexpired ANS tape may be written on through a non-ANS data control block; all ANS tapes must be initialized by the Label processor; no tape serial number specification is allowed at the operator's console; specification of an output serial number in an ANS data control block forces processing to be done only on a tape already having that serail number; tapes mounted as input tapes may not be written; and tapes mounted as other than input tapes must have a write ring.

2. Semi-Protective Mode: A mode of tape protection in which a warning is sent to the operator when an ANS data control block attempts output on a non-ANS tape or an unexpired ANS tape, when a non-ANS data control block attempts output on an unexpired ANS tape, or when a tape mounted as available for input and output has no write ring. The operator can authorize the overwriting of the tape or the override of the input and output specification through a key-in. ANS tapes may be initialized by the Label processor or may be given labels as the result of an operator key-in; tape serial number specification is allowed at the operator's console; and specification of an output serial number in an ANS data control block forces processing to be done only on a tape already having that serial number unless the operator authorizes an overwrite.

Before closing the discussion of file security, a few words should be said about security when the Extended Data Management System (EDMS) is used. Security under EDMS necessarily goes further than the standard file system. The reason for this is that with EDMS the data base is by definition an integrated one that will in all probability require multiple users in multiple accounts to access all or part of the information. The advantage of EDMS is that it permits this type of usage while providing numerous security checks to prevent unauthorized access.

Some of these security checks include

- Ability to segment the definition of the entire data base (schema) into logical sub-sections (sub-schemas). This allows access permission to only limited areas of the data base for certain users.

- EDMS requires separate passwords to authorize modification of a schema, generation of a new sub-schema, use of an existing sub-schema, read access to specific data groups or items and update access to specific data groups or items.

- Data enciphering/deciphering logically combines a user-selected bit pattern with data as it is entered into the database. Each successive word in an EDMS page is automatically modified by its predecessor before it is stored. Secure data can therefore be protected even if access to the group or item containing the data has been achieved, since the key pattern must be supplied by all users who access the area before the actual data can be regenerated. Memory dumps of the database do not contain the key pattern.

Thus the file systems supported under the CP-V system provide numerous protection vehicles. While they can be used to absolutely prevent access to files, they also have the flexibility to permit access that is limited by any combination of account, function and/or areas of files.

## SHARED LIBRARIES

A shared (public) library is a collection of frequently used subroutines which is treated by CP-V in such a way that multiple programs may simultaneously use the same copy of the library. This results in efficient use of main memory, swapper space, and bandwidth. A number of public libraries are supplied with CP-V (e.g., the FORTRAN Run-Time Package and the COBOL library). User installations may create additional public libraries which suit their specific requirements.

## SHARED PROCESSOR FACILITIES

The shared processor facilities of CP-V permit the sharing of the code for compilers, assemblers, command language processors, debuggers, libraries, and other programs among all simultaneous users. Each user of a shared processor has his own copy of only the data and DCB portion of that program; the procedure (code) portion is shared by all users associated with the shared program. Furthermore, the shared processors are prepackaged for immediate loading and their secondary storage disk address placed in CP-V main memory tables. This makes for very fast access through the swapping mechanism.

Shared processors are not limited to company-supplied programs. The facilities may be effectively used whenever a program has a high probability of common usage. Service bureaus, for example, may use the mechanism for proprietary packages. Corporate installations may use the mechanism for programs with a high use frequency.

To qualify as a shared processor, a program must meet certain requirements. Three of the more important of these requirements are as follows:

1. Shared processors are allowed only one level of overlay. There is no restriction on the number of overlays but only one of them can be associated at a time.

2. Data cannot be included in overlays; it must be in the processor root.

3. Shared processors are often written in assembly language but may be written in FORTRAN or any processor providing pure code output. Shared processors written in FORTRAN must be initialized by some assembly language code that associates the FORTRAN library and links to the FORTRAN code.

These requirements are outlined in greater detail in the CP-V/SP Reference Manual, 90 31 13.

# CHAPTER 4. USER-DEVELOPED PROGRAMS

## OVERVIEW

A user creates, compiles, loads, and executes a program in the following manner:

1. Source language programs are created on punched cards or via the Edit processor in the time-sharing mode.

2. The program is assembled or compiled by calling the appropriate processors. In the time-sharing mode, the processor is called with a TEL command such as:

   ```
   !ANSF THEIRPROG.1286A ON NEWFILE
   ```

   In this example, the ANS FORTRAN source code resides in a file named THEIRPROG in account 1286A. The object code is to be stored in a file called NEWFILE in the user's account. After the command has been entered, the FORTRAN compiler prompts for compilation options. When the user has entered the compilation options, the compilation begins. This general approach is used by all CP-V compilers and assemblers in the time-sharing mode. Incremental compilers such as BASIC and interpreters such as APL eliminate some or all of the following processing steps, including them (or their equivalents) automatically - i.e., without explicit user command.

   In the batch and remote processing modes of operation, the program is assembled or compiled by calling the appropriate processor with a system control command such as

   ```
   !ANSFORT LS,LO,PS
   ```

   In this case, the compilation options are specified on the command which calls the compiler. The whereabouts of the source code and the disposition of the object code are specified on ASSIGN commands which precede this command.

   The output of the assembly or compilation is a relocatable object module (ROM).

3. The ROM or a set of associated ROMs is loaded by either the Load or LYNX load processor. These processors combine ROMs into a single entity called a load module (LM).

4. Program execution is initiated by the RUN command. The batch-mode RUN command has a different format than the on-line RUN command. However, either RUN command may be used to execute a load module formed by either of the two load processors. The START command is also available to on-line users for execution of load modules.

Time-sharing users may execute a program under the control of a debugging processor to facilitate the location and correction of errors.

Some of the more specific details of user-developed programs are given in the following sections.

## PROGRAM LOAD AND EXECUTION

There are two processors that can be used to control loading and execution of object programs: the Load processor and the LYNX processor.

Load is a two-pass overlay loader. The first pass processes not only ROMs but previously formed load modules or a combination of both. (For example, Load processes dummy sections from library load modules as well as from ROMs.) The first pass also processes expressions for definitions and references (primary, secondary, and forward references). The second pass forms the actual core image and its relocation dictionary. Load is available only in the batch mode.

LYNX has all of the capabilities of the overlay loader, Load, and also provides control over internal and global symbol table construction. LYNX is in fact a preprocessor for the Load loader. After it analyzes the user's commands, it constructs a table of loader control information which it then passes to the Load loader. It is Load which actually performs the loading process. LYNX is available in both the batch and on-line modes.

## PROGRAM OVERLAYS

An overlaid program is one that has only one segment resident in main memory permanently. The other segments are called for by the

M:SEGLD procedure call and brought in as needed. They may reside (at different times) in the same main memory area, thus reducing the amount of main storage required to house the entire program.

If a program is to be overlaid, a TREE control command must be the next control command following the command which called the Load or LYNX processor. It must specify the overlay structure of the load module to be formed, so that the logical segments of the program will be loaded from secondary storage into main memory as required. It is the user's responsibility to plan the relationship of these segments.

The relationship of the segments that comprise an overlay program can be represented graphically by means of a tree diagram, as in the example shown below. The horizontal coordinate of the diagram denotes increasing main storage (address) allocation, from left to right. The vertical coordinate denotes overlays. The leftmost segment, or "root", is that portion of the program that resides in main storage through program execution. A "path" of an overlay consists of those segments that may occupy main storage at the same time. The portion of a path that extends from the start of the program (i.e., the root) to a given segment is termed the "backward path" of that segment.

The example in Figure 4-1 consists of four paths, any one of which may be present in main storage at any given time. Segment A, below, is the root of the program and is never overlaid by another segment. Any path may be loaded into main storage and overlaid as many times as required by the program. All segments of the load module are saved in disk storage and, when a segment that has been overlaid is called again by the executing program, the original copy is loaded from the disk. Therefore, any communication between two overlay segments (e.g., D and E, below) must be done in a part of the backward path common to both. Although the tree below is singular, most actual overlaid programs consist of two parallel trees, one for data and one for program. Both are fetched by a single segment load call.

```
!TREE  A-(C-(D,E),B-(G,F))
```

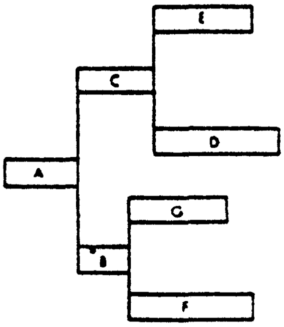


Figure 4-1. Sample Tree Structure

## LIBRARIES

The purpose of a library is to collect frequently-used routines in a form that expedites their inclusion into user programs. The loader associates a library routine (with the program it is contructing) on a conditional basis - he condition being that the library routine contain a DEF that satisfies an undefined PREF in the main program. The loader accomplishes this task by performing a search of the library to locate the DEF.

A library may contain modules of the following types:

- library load module - constructed by Load, LYNX, or LEMUR.

- concatenated ROM module - constructed by LEMUR.

(A library may contain one or more of either or both types of modules.)

The library load module is advantageous because the routines are converted to a format approximating the final user program main memory image. When the loader forms a user load module, these library routines (already in load module format) are processed in an expeditious manner. However, there are restrictions on library load modules. In particular, all control sections within the module must be of uniform protection type.

The ROM type of library load module presents no such restrictions. Any ROM or set of ROMs may be entered into the library by the LEMUR processor.

A library is a named, keyed file. The reason library routines are placed in one file is to minimize the number of OPENs and CLOSEs the loader must perform to include several library routines in a user program. A library file is created and named by the Load, LYNX, or LEMUR processor when appropriate options are specified for the processor. There are virtually no restrictions on the number of libraries which may reside in an account.

Internally, a library consists of two types of records - dictionary records and module records. A dictionary record has the TEXT name of the DEF as its key. The dictionary record itself contains the TEXT name of the library module in which the DEF is defined. A module record has as its key the module name appended by other information (depending on whether it is a library load module or a concatenated ROM file). The module records are the actual load module records or the ROM module records.

Library modules may be individually deleted and copied. New modules may be added and existing modules may be replaced. These functions are available through the LEMUR processor.

## MEMORY PROTECTION

Monitor pages and unallocated virtual pages are protected against access by user programs with the map access protection provided by the CP-V hardware. The access protection types are

| | |
|---|---|
| 00 | read, write, and execute access permitted (data). |
| 01 | read and execute access permitted (pure procedure). |
| 10 | read access permitted (static data). |
| 11 | no read or write permitted (no access). |

Programs that either deliberately or inadvertently access the monitor (by reading it or branching into it) will trap. The same restriction also applies to other areas of the machine that were not owned by the program (e.g., read access to unobtained common or dynamic data pages). The first page of main memory is an exception to these rules; its access is always set to read only.

## VIRTUAL MEMORY LAYOUT

The user's 96K words of virtual space are divided as follows:

1. 8K words for monitor overlays and user context (JITs and buffers).

2. 88K words for user procedure, DCBs, and data unless the user program requires the use of a special shared processor or a public library. In this case, the user area is 72K words and the special processor area is 16K words.

With the exception of a fixed minimum requirement of six pages for monitor overlays, one page for JIT, and three pages for the file buffers, the 96K words of user area is demand allocated.

The LYNX and Load loaders place ROM data, including any data overlays, in memory beginning at 40K then directly follow this with the DCBs, procedure, and procedure overlays. When a BIAS is specified, the load module is created at the specified location even though it may not be possible to run the load module there.

Load modules are constructed from ROMs composed of control sections. A control section is of type 00, 01, or 10. All control sections of type 00 are gathered together by the loader and designated as DATA. Similarly, all control sections of type 01 and type 10 are gathered together and designated as PROCEDURE and STATIC DATA, respectively.

Except for DCBs, DSECTs or CSECTs with value 2 or 3 are changed to 1. That is, no-access and read-only data are loaded with pure procedure. Any DSECT that has a name beginning with M: or F: is assumed to be a DCB and is removed to the DCB area and listed in the DCB table.

DCBs and the DCB name table are allocated in the user context area (10 protection) rather than in the root procedure area (01 protection).

Internal symbol tables are generated for use by a debug processor (e.g. Delta) if a program is assembled with the appropriate option. An internal symbol table is built for each load module and is included in the load module as a keyed record consisting of the element file name appended with an X'10'. A symbol table can be loaded by a debug processor for an overlay or a nonoverlay program by specifying its element file name. If the element file contains more than one ROM then only the symbol table for the last ROM is produced. A symbol table that is generated during a load from the GO file cannot be accessed by a debug processor. No internal symbol tables are generated for library load modules.

Figure 4-2 shows the layout of virtual memory for a program loaded by Load or LYNX. Ordinary shared processors follow this layout. Figure 4-3 shows the actual background memory layout at execution time.

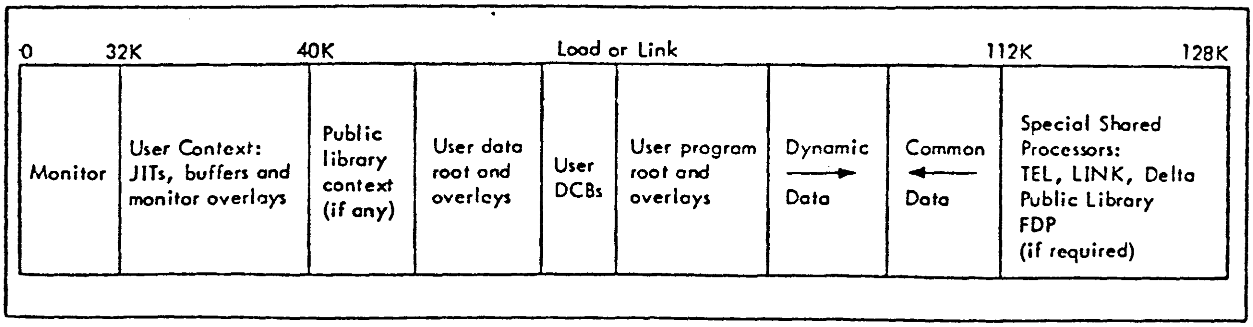
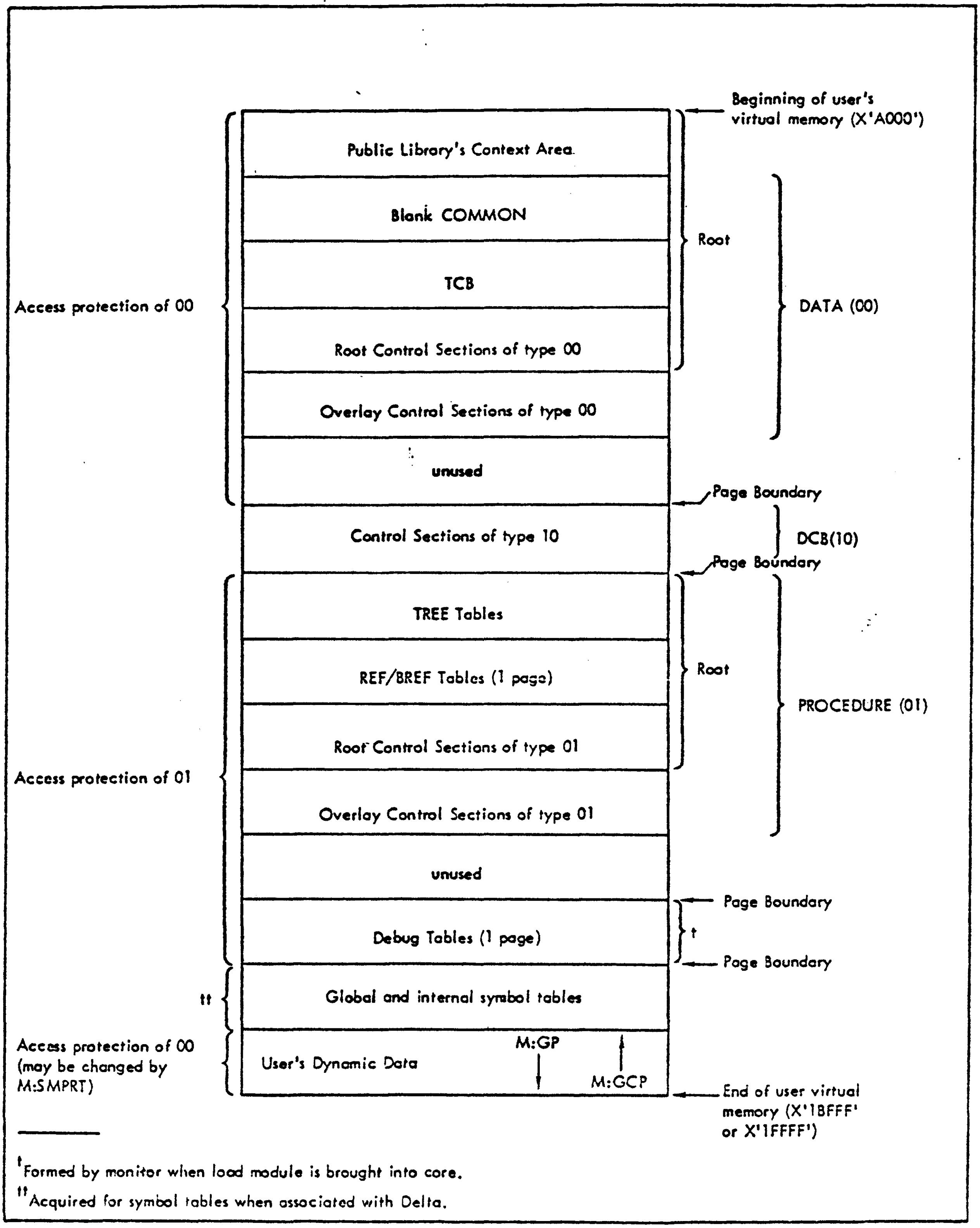

0
32K
40K
Load or Link
112K
128K
Monitor
User Context: JITs, buffers and monitor overlays
Public library context (if any)
User data root and overlays
User DCBs
User program root and overlays
Dynamic → Data
Common ← Data
Special Shared Processors: TEL, LINK, Delta Public Library FDP (if required)

Figure 4-2. Virtual Memory Layout


Beginning of user's virtual memory (X'A000')
Public Library's Context Area
Blank COMMON
Root
TCB
Access protection of 00
DATA (00)
Root Control Sections of type 00
Overlay Control Sections of type 00
unused
Page Boundary
Control Sections of type 10
DCB(10)
Page Boundary
TREE Tables
REF/BREF Tables (1 page)
Root
PROCEDURE (01)
Root Control Sections of type 01
Access protection of 01
Overlay Control Sections of type 01
unused
Page Boundary
Debug Tables (1 page)
†
Page Boundary
††
Global and internal symbol tables
Access protection of 00 (may be changed by M:SMPRT)
User's Dynamic Data
M:GP
M:GCP
End of user virtual memory (X'1BFFF' or X'1FFFF')

[†] Formed by monitor when load module is brought into core.

[††] Acquired for symbol tables when associated with Delta.

Figure 4-3. User Virtual Memory Layout, Load and LYNX Processors

# CHAPTER 5. FILES AND DEVICE INDEPENDENT I/O

## INTRODUCTION

CP-V provides all I/O services through a common set of services. Programs may be written without the need for explicit knowledge of the file or device to which I/O will actually take place. Selection of the file or device can be done internally to the program (M:OPEN) or externally via a control command (SET or ASSIGN). In addition, a set of default device assignments are provided which make appropriate device selections for batch or on-line jobs.

All requests for I/O services specify a data control block (DCB) name for use in performing the I/O. The DCB is the storage for maintaining the actual I/O connection for the user as well as as the repository of information concerning results of an I/O operation, etc. Figure 5-1 illustrates the various connections established for performing I/O. The following points are keyed to the connections illustrated in the figure.

1. The user program references a DCB via the CAL,FPT mechanism.

2. The DCB is connected to one of several types of I/O facilities via M:DCB, !SET, !ASSIGN, or M:OPEN.

   a. FILE - either a file in public storage or on a private pack named as a resource.

   b. CP-V or ANS labeled tape - a file on a labeled tape named as a resource.

   c. DEVICE, physical device type - causes connection to a private device (foreign tape) mentioned as a resource, a symbiont file (card reader, line printer, etc.), or to the user terminal.

   d. DEVICE, operational label - causes a "functional" connection ("the place where all listings go") assigned by management to the appropriate default device: either symbiont file or user terminal as established by the installation for on-line or batch.

   e. DEVICE, logical device stream - causes connection to a symbiont file for the local or remote device as established by the LDEV command.

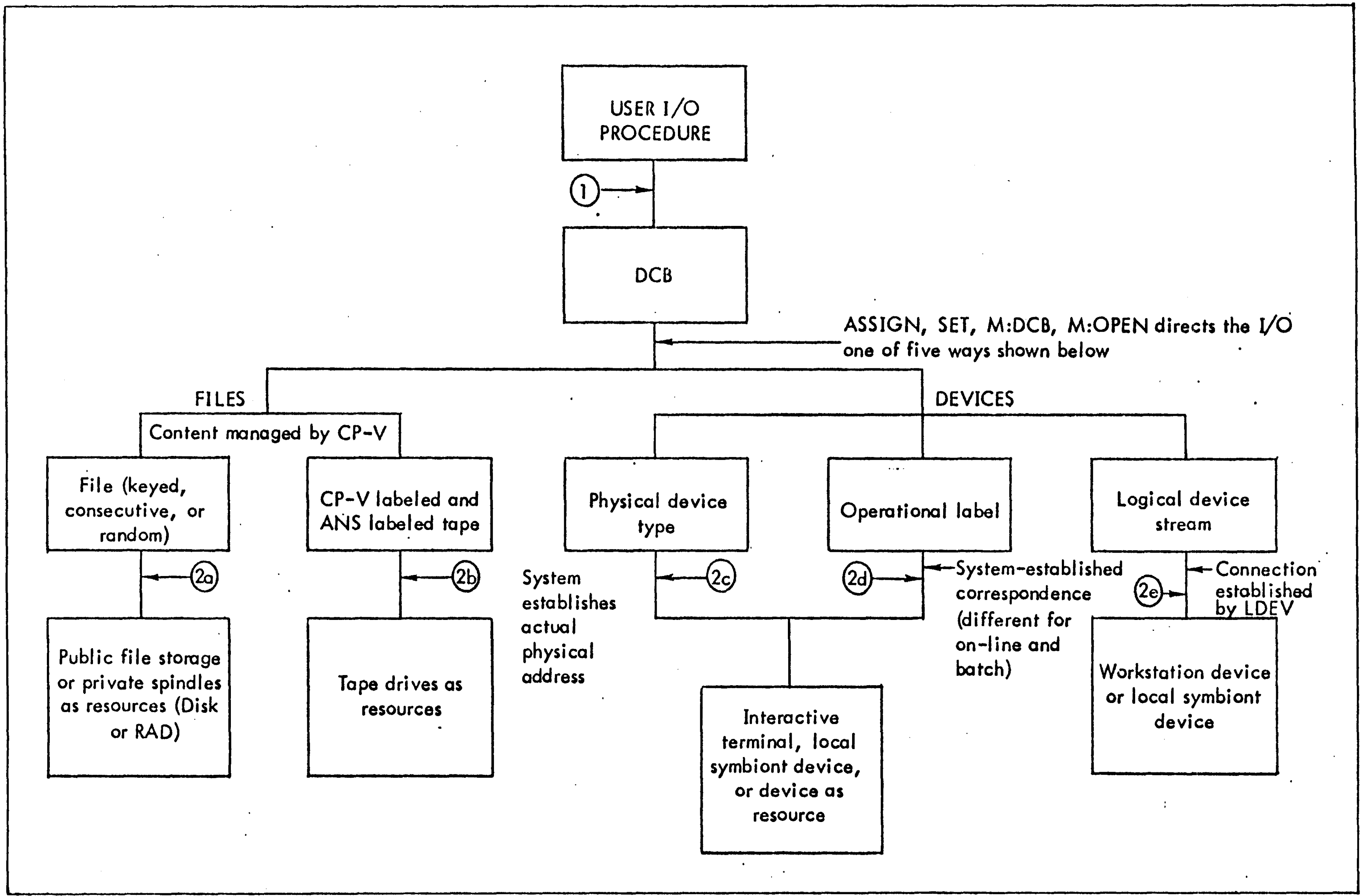

USER I/O PROCEDURE
1
DCB
ASSIGN, SET, M:DCB, M:OPEN directs the I/O one of five ways shown below
FILES
DEVICES
Content managed by CP-V
File (keyed, consecutive, or random)
CP-V labeled and ANS labeled tape
Physical device type
Operational label
Logical device stream
2a
2b
2c
2d
2e
System establishes actual physical address
System-established correspondence (different for on-line and batch)
Connection established by LDEV
Public file storage or private spindles as resources (Disk or RAD)
Tape drives as resources
Interactive terminal, local symbiont device, or device as resource
Workstation device or local symbiont device

Figure 5-1. Connections Established for Performing I/O

For convenience, a number of DCBs are available with default assignments to the operational label of the same name. These are set up for common system usage, e.g., LO = listing output and SI = symbolic input are the assignments of the M:LO and M:SI DCBs.

Each request for I/O service from the monitor is made by inclusion of an I/O call in the user's program. This call generates a Function Parameter Table (FPT), which in turn refers to a Data Control Block (DCB). The combination of the I/O call, the FPT and the DCB provides the information that the monitor needs to perform the requested operation.

Generally, the DCB contains the kind of information that is specific to a device (e.g., for output to a line printer, number of lines per page is one value in the DCB). The FPT contains a far smaller set of information that is specific to the operation to be performed (e.g., the location and size of the buffer that is to be output to the printer in this specific operation). Separation of information into the DCB and the FPT allows the user to create one DCB for a type of I/O and reference that DCB throughout his program, whenever he requires that type of I/O. Each time that he references that DCB, he generates an FPT with the specific information required for that particular I/O operation.

I/O procedures are provided for the following I/O functions:

1. <u>File Maintenance</u>

   Create a Data Control Block

   Open a Data Control Block

   Close a Data Control Block

   Set Error or Abnormal Address

   Check I/O Completion

   Declare Temporary File

2. <u>Data Record Manipulation</u>

   Read a Data Record

   Write a Data Record

Delete a Data Record

Truncate All Blocking Buffers

3. File Manipulation

Position n Records

Position File

Close Volume

Rewind

Write End-of-File

Insert or Delete a Symbiont File

4. Device Manipulation

Set Listing Tabs

Skip to Top of Form

Set Number of Printable Lines

Set Line Spacing

Specify Direct Formatting (Transparent, FBCD)

Specify Vertical Format Control

Specify Page Count

Change Output Form

Change Device Mode or Record Size (Binary, BCD, Packed, Unpacked)

Specify Beginning Column

Specify Output Header

Specify Card Punch Sequencing

Determine Number of Lines Remaining

Check Correspondence of DCB Assignments

## FILES

A general understanding of files and the way that the monitor deals with them will help the user to obtain the high level of performance available.

A file is an organized collection of information. This collection of information may consist of one or more programs, one or more sets of data, or some combination of programs and data. Under CP-V, a user always accesses files through the monitor - never directly. An option does exist, however, that allows a user to deal with a file (e.g., a non-standard set of data on an unlabeled magnetic tape) as though he were accessing it directly.

The monitor maintains a directory of accounts having files maintained between jobs. This is called an Account Directory, and contains, with each account number, an address of a directory of files (termed a File Directory) for that account. A File Directory contains, with each file name, an address of a table containing file attributes and disk locations for that file. The table is called a File Information Table. To summarize, the monitor has a single Account Directory, which in turn points to a File Directory for each account. Each File Directory, in turn, points to a File Information Table (FIT) for each file.

Each file has associated with it (in the FIT) information controlling who may access the file and how it may be accessed. This information can include both a password and a list of which accounts may execute, read, or update the file. To access a file, a user must be running under an account which is authorized to access the file and provide the proper password (if one exists). In addition, to access control information, the FIT also contains the file's creation date, date of last modification, date of last access, and expiration date.

## FILE IDENTIFICATION

In CP-V, files are identified according to a standard format. The identification is assigned to the file by the user when the file is created. The format for a file identifier (fid) is one of the following:

```
    name
    name.account
    name..password
    name.account.password
```

where

name is the name of the file and may have a maximum of 31 characters. (Certain processors in CP-V have shorter length restrictions.)

account is the account number of the file and may have a maximum of eight characters. A user normally may not create a file in any account other than the one under which he is running. He may not execute, read, or modify a file in another account unless he has been given read or write access to that file.

password is the password for the file and may have a maximum of eight characters. A password is useful in preventing unauthorized users from accessing the file.

The various combinations for a file identifier are as follows:

name file in the user's account. The file does not have a password associated.

name.account file in a specified account. (It could be the user's account, but this format is required only if the account is other than the user's account. The file does not have a password associated.)

name..password file in the user's account with the specified password.

name.account.password file in the specified account with the specified password.

## FILE FUNCTION AND DISPOSITION

A file may be opened with one of four functions: two of these, IN or input and INOUT or update, are used to access a file which had existed prior to this open; the other two, OUT or output and OUTIN or scratch, are used to create a new data aggregate which had not existed prior to this open. There are three possible specifications for the file disposition option: REL or release, SAVE, and JOB. Any one of the three may be specified at open time and either REL or SAVE may be specified at close time. Now, let us consider the impact of each of these options and the several significant combinations of them.

It has been noted that to create a new file one must specify OUT or OUTIN. If the REL dispostion option is used or implied (see below) with such an open, it indicates that the file to be created must be released when it is closed; thus, it is an obvious error to combine OUT with REL, and such an open is rejected. The other combination, OUTIN with REL, results in a true scratch file which is never to be entered into the file directory and thus has no identification other than the device control block with which it is associated. Storage space requirements for such a file are accounted for against the user's temporary granule authorization.

If a file opened OUT or OUTIN is closed with an explicit specification of SAVE, it will be entered into the file directory unless the open process failed to explicitly specify SAVE or JOB, in which case the file is unconditionally released at close time and the file directory contents are not altered. If an explicit SAVE specification is not made when an OUT or OUTIN file is closed, again the file is released and the file directory contents are not altered. Note that when a job step is completed, all open device control blocks are closed with no explicit disposition specification, and so all open output files are released at that time. The only exception to this is M:DO which is closed with explicit SAVE in order to ensure that diagnostic output will be received.

Now, let us consider the cases of files opened OUT or OUTIN with either SAVE or JOB disposition. All such combinations indicate the intent to create a new file which will probably be entered into the file directory. (See the above paragraph for a discussion of how to overcome this intent.) Unless a job is executing at a high privilege level of X'CO' or greater, it cannot create a new file in an account other than the one under which the job is logged on with one exception. If a file already exists with the same identification as that desired for the new file; if, further, the already existing file permits WRITE access to the user in question; and if, finally, the already existing file is not currently open, then the user may create a new copy of such a file. When an OUT or OUTIN file with SAVE or JOB is closed with explicit SAVE, the name is entered into the file directory; and any previously existing file with the same identification is released. In addition, if JOB has been specified on the open, the file identification is given the same treatment as though it had been mentioned in a M:TFILE procedure call. All files which have had their identification mentioned in such a procedure call are released when the creating user logs off. A JOB file may only be accessed by the creating user or a user with at least X'CO' privilege. Storage space requirements for JOB files are again accounted for against the user's temporary granule authorization. All storage space requirements for files other than true scratch or JOB files are accounted for against the user's permanent granule authorization.

The file disposition option at open time for input and update opens is essentially insignificant and the disposition is completely controlled by the specification on the close. (There is a name substitution option available for locating JOB files which is only operative in the event of explicit JOB disposition specification at open time.) If the specification is explicitly REL, the file is released and the identification is erased from the file directory; otherwise, the file is retained and no change is made to the file directory. When an existing JOB file is reopened in the update mode, the disposition in the device control block is forced to JOB so that granule accounting may be correctly handled.

## FILE ORGANIZATION

The information in a file may be structured in one of three ways. It may be a keyed, consecutive, or random file. The type of structure is called the organization of the file and is a file attribute. The information in a file may be accessed in one of two ways: direct (unique identifier of record supplied) or sequential (using the ordering relationship of records). Thus file access is simply the way in which a file is being accessed at a particular instance of usage.

The following chart illustrates the allowable combinations of organization and access:

| Organization<br>Access | Keyed | Consecutive | Random |
|---|---|---|---|
| Direct | X |  | X |
| Sequential | X | X | X |

Note that other devices behave much like consecutive files (records have no names and are order dependent) and only sequential access is allowed.

# KEYED FILES

Keyed files are those in which each record has an identifying key associated with it. A key consists of a byte string, the first byte of which states the number of bytes in the string. The contents of each byte may be a binary number or a character. A key may consist of up to 31 bytes plus a count byte.

As the file is being created, a master index is also created with an entry for each keyed record in the file. The keys are sorted into collating sequence so the file can subsequently be accessed sequentially. The entry contains such information as the key, disk address of the record, size of the record, and position of the record within the blocking buffer.

The records are automatically packed into blocking buffers with the last portion of the last record extending into another buffer as necessary. If the record is large, it is written directly from the user's area instead of being packed into a buffer. Keyed files may be accessed by direct or sequential access.

Keyed files have a multilevel index structure to provide fast direct access. A multilevel index sturcture is a collection of hierarchical levels of index blocks, where the entries in a higher level point to index blocks at the next lower level and the entries in the lowest level (called level 0) point to data records. This is best illustrated by the hypothetical example shown in Figure 5-2. Unless specified otherwise by the user, the multilevel structure is initially built when the file is closed if the file has more than three level 0 index blocks.

In the example shown in Figure 5-2, the keyed file has:

- 31,150 records and the keys at level 0 point to these data records. Based on an 11-byte maximum key length, there are 80 keys in each level 0 block and 127 keys in each higher-level block.

- 390 index blocks at level 0, four index blocks at level 1, and one index block at level 2. The next higher-level is built if the last level has more than three index blocks.

This example shows the data in the same sequence as the keys. This need not be true and will not be true after random addition of records.

Each entry in a higher-level index block contains the disk address of an index block at the next lower level, and the key of the first key in that block.

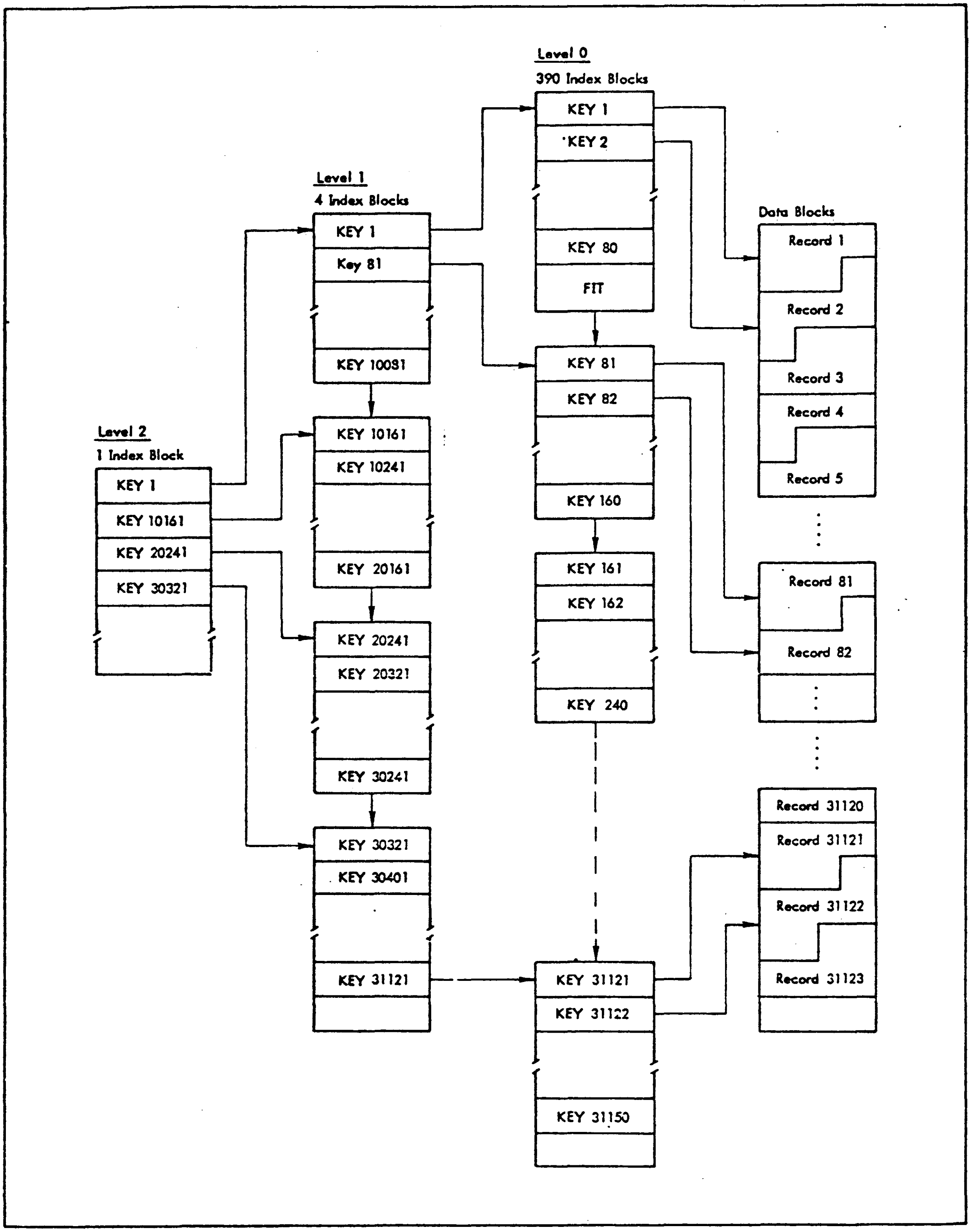


Figure 5-2. Example of Multilevel Index Structure

The multilevel index structure can considerably speed up the direct access of a large keyed file, at only a small cost of secondary storage space. Since the keys are ordered in ascending sequence, at most it would take three index block accesses to locate a data record as shown in the example. Wihtout the higher-level structure, it could take up to 390 index block accesses.

The user has control over the initial creation of the multilevel index structure and he can specify when and if the higher-level structure should be rebuilt. This can be specified by using the NEWX option on the ASSIGN control command or the M:OPEN and M:DCB procedures.

## CONSECUTIVE FILES

Consecutive files are files whose records are organized in a consecutive manner; i.e., the user is aware of no identifying keys associated with the records. The records may only be accessed sequentially.

The principal benefit of consecutive files to system operation is a reduction in the amount of space required for the files on disk and RAD and a consequent reduction in the time required to traverse the files. Also, there is no need to identify each record.

All position operations for consecutive files are done with user selected system procedures rather than with I/O operations. The positioning is only effected when a data transfer operation is about to take place. At that time, there will be three known points in the file that can be used as a starting point: beginning of file, end of file, and the position reflected by the DCB (see DCB in glossary). The starting position chosen will be the one that requires the fewest record skips to be made.

## RANDOM FILES

Random files provide a basic organization for those users desiring to manage their own files. Random organization differs from keyed and consecutive organization as follows:

1. A random file is simply a collection of contiguous granules on the specified device type. The number of

granules is specified at the time the file is created (and may not be expanded after it has been created). If the requested number of granules are not available contiguously, an abnormal code is returned to the user and the file is not opened.

2. The user must specify a relative starting granule number with each read or write and a byte count (a default byte count may be used).

3. Each write/read consumes the entire specified granule. The contents of the granule include no system information. Management of the user's data is the responsibility of that user.

Thus, the monitor provides allocation of granules, security checks and normal I/O queuing service and clean-up. The user is responsible for record management.

## FILE ACCESS

Records may be accessed within a file by either of two means, direct or sequential access. The interaction of the type of access used for a given operation and the mode in which the file is opened results in some rules, or limitations. These rules are listed below for each type of access and each mode in which a file may be opened.

## DIRECT ACCESS

For consecutive files, sequential access is assumed. For keyed files, the following rules apply.

### OUTPUT FILES (OUT)

When a WRITE is given, a key must be specified. The keys do not need to be given in a sorted order. They will be ordered as they are stored on disk.

Unlike sequential output files, a WRITE never causes forward information to be deleted.

Reading is not allowed.

## SCRATCH FILES (OUTIN)

A scratch file is identical to an output file, except that reading is permitted before the file is closed. As for output files, a key must be specified on each WRITE. The keyed record is merged into the file.

A READ may or may not specify a key. If a key is specified, a search is made of the file until the key is found and the record is then read. If the key is not found, an error return is executed. If a key is not specified, the next sequential record is read.

The FWD and REV options apply on read operations not specifying a key. If a key is specified, these options are ignored. PRECORD (positioning) operations are performed in the same way as for sequential output files. A WRITE does not cause forward information to be deleted. A READ before the first WRITE returns an error code.

## INPUT FILES (IN)

Records may only be read; writing is not allowed. The READ function is the same as that for scratch files. PRECORD operations are allowed.

## UPDATE FILES (INOUT)

The READ function is the same as for scratch files. PRECORD operations are allowed.

The WRITE function may or may not have a key specified. If a key is not specified, the WRITE function must have been preceded by a READ. If it is, the record just read is updated; if not, an abnormal code is signaled.

Write operations may indicate whether the record is intended to be a new or replaced record. The absence of an option indicates replacement. NEWKEY indicates a new record. ONEWKEY indicates the record may be new or replaced.

The DELETE function may be used. If a key is specified, a search of the directory is made to find the specified key. The record is then deleted. If a key is not specified, the DELETE operation must have been preceded by a READ, and the key just read will then be deleted.

## SEQUENTIAL ACCESS

Sequential access may be used when accessing records with keyed or consecutive organization.

## OUTPUT FILES (OUT)

When a file is opened in the OUT mode, records may only be written; reading is not allowed. If the file has been declared a keyed file, a key must be given with each write operation and this key must be a new key (i.e., it must not have been used before). If the key has already been used, no information is written and an abnormal code is returned. The keys must be given in a sorted order. For example, if the user writes records with keys A, C, and D, respectively, and then writes a record with key B, the record will not be written and an error return will be executed.

The PRECORD FWD (position record forward) and PRECORD REV (position record backward) operations are allowed on both keyed and consecutive files. A BOF code is given when the beginning-of-file is reached, and an EOF abnormal is given when the end-of-file is reached. Otherwise, for keyed files, the pointer to the current entry in the master index is decremented or incremented. For consecutive files, a directional count of records to skip from the current position is established. Positioning will not occur until the next read, write, or delete operation. A WRITE operation following PRECORD causes all forward records to be deleted.

## SCRATCH FILES (OUTIN)

The same rules that apply to output files also apply to scratch files, except that reading is allowed, following a write. Reading may be directional; either forward or reverse. A READ with REV implies that the record preceding the current position is to be read. If no direction is specified, FWD is assumed. A READ order issued prior to the first WRITE will result in an abnormal return.

When reading a keyed file, a key may or may not be specified. If a key is specified, a search is made for the specified key. The FWD and REV options are ignored when a key is specified. If a key is not specified, READ FWD implies that the next record in sequence is to be read. READ with REV implies that the record immediately preceding the current record is to be read. Whenever a keyed file is read, the KBUF field of the DCB contains the address at which the key of the record just read is stored.

Reading a consecutive file is the same as reading a keyed file without specifying a key.

A WRITE deletes all forward information.

## INPUT FILES (IN)

This is the same as for direct access input files.

## UPDATE FILES (INOUT)

For a keyed file, this is the same as for direct access update files. For a consecutive file, a WRITE deletes all forward information.

## SIMULTANEOUS FILE USAGE

The SHARE mode feature extends the use of keyed and random files by permitting simultaneous access to a file by up to 127 updaters <u>and</u> up to 127 readers. Thus several user programs executing concurrently in separate jobs may be generating reports from a data file while other user programs are concurrently modifying data items within the file.

Responsibility for coordinating concurrent update activity is divided into two parts, one controlled and provided by the operating system and the other by the application programs via the system's enqueue/dequeue service. The operating system guarantees the physical integrity of the file so that it remains properly connected regardles of the update activity and also ensures that readers are provided with the most up-to-date information in response to their requests.

Coordinating and guaranteeing logical integrity of the file (primarily the data content) is the responsibility of the application programs, since for the keyed file organization any connection of the data in one record of a file with that in another record of the same or another file is carried in the application program, not in the file itself. A single example will serve to illustrate this.

Suppose that a file contains records recording a parts inventory - each containing the available number of bolts, washers, nuts, etc., in various sizes. Without any special coordination, the number of any given item can be determined by querying the file even in the face of additions and removals by a concurrent updater. If, however, the application needs to first determine the available number and then remove a quantity from stock, then the record must be locked between the read and the update to preclude the possiblity of the stock being taken by another updater.

More elaborate record locking requirements may exist depending on the application. For example if a fastner must be made up of a bolt, a nut and a lock washer, then these three records must be acquired and locked prior to making the needed updates.

Applications use the system's enqueue/dequeue facility to gain exclusive access to the records. Enqueue/dequeue is a generalized service and guarantees exclusive or shared access to named items as required and requested. It is the responsibility of all users of the service to agree on the meaning of the names - for example the names of the records containing inventory count of nuts, bolts, and washers.

## FILE STORAGE DEVICES

The three general types of storage media available for user files are (1) disk, (2) labeled magnetic tape, and (3) other physical devices (e.g., cards, unlabeled magnetic tape, etc.).

## DISK STORAGE

Both RAD and disk pack devices are used for secondary storage. Any combination of these devices can be defined at SYSGEN time. A disk pack device has dismountable volumes and can be declared either a public or private device at SYSGEN time, while a RAD device, not having dismountable volumes, can only be declared a public device.

A public disk pack device has only one volume that can be recognized by the monitor and that volume must be mounted at all times while the system is active.

A private disk pack device has any number of dismountable volumes that can be recognized by the monitor. The operating system requires only that those volumes needed for execution of the user's job be made available and be mounted.

## STORAGE ALLOCATION UNITS

For allocation purposes a disk pack device is partitioned into logical units, either granule or cylinder. RADs are partitioned and allocated in granule units only. A granule unit equals 512 words and is equivalent to two sectors.

## FILE ALLOCATION

Keyed and consecutive file space is allocated on a demand basis as the file is being created or updated. Therefore the file does not necessarily exist in contiguous areas on a RAD or disk pack device and can exist on many different physical devices. Random file space is contiguous and is allocated when the file is opened.

A public file resides on a public device (RAD and/or disk pack); a private file resides on private disk pack volumes. A public file can be allocated in granule or cylinder units; a private file is always allocated in cylinder units.

<u>Files on Public RAD and Disk Pack.</u> Allocation of space for files on RAD and/or disk pack follows a set of rules that may be

altered and controlled by both the user for individual files and by the system manager on an account or system-wide basis. The scheme provides for best system performance, in absence of specification by the user or system manager, or for good performance of individual jobs by careful selection of disk pack or RAD to optimize the program's performance.

Although the rules stated below control the preferred allocation, the system will continue to look for space on other devices on request as long as the user-allowed limit is not exceeded and the space physically exists.

In the absence of other specifications, the monitor uses the following rules to determine the placement of files on RAD or disk pack:

1. All permanent files (opened OUT or OUTIN with SAVE or INOUT) prefer disk pack.

2. All temporary files (opened OUT or OUTIN and RELease) and all job files prefer RAD.

3. All account directories (AD) and file directories (FD) prefer RAD.

4. All star files (system temporary files for ROMs, LMs, debuggers, etc.) prefer RAD.

The system manager can control file space allocation. For example, the system manager may separately limit the amount of space on RAD or on disk pack available to an individual user.

A user program or job may control the allocation of files to RAD or disk pack using either SET or ASSIGN control commands or the M:OPEN program procedure.

Public Random Files. A public random file is allocated on a public device of the type specified, either RAD or disk pack. If disk pack was specified, the monitor attempts to allocate in cylinder units before allocating in granule units.

Private Files. All the index and data blocks of a keyed or consecutive private file are allocated from one or more private disk pack volumes. A keyed, consecutive, or random file can extend beyond volume boundaries.

## RECORD BLOCKING

The system will automatically block records for keyed and consecutive files in 512-word blocks to provide more efficient use of disk space. The user has no knowledge of this blocking and, when reading, will receive the appropriate record within the block and not the entire block.

When updating a keyed file, the user may rewrite a record in a size larger or smaller than the original record size. If necessary, the monitor will allocate additional disk space to accommodate the larger size.

A write with a 0 byte count to a keyed file will result in a master index entry for the record with fields in the entry pertaining to disk address, record size, and displacement into the blocking buffer all set to zero. A write with a 0 byte count to a consecutive or random file will be ignored.

## LABELED TAPE

CP-V handles two types of labeled tape, CP-V labeled tape and ANS labeled tape.

## CP-V LABELED TAPE

A CP-V labeled tape is given standard CP-V labels when I/O is first performed on the tape. No tape initialization is required.

For labeled tapes, record blocking is performed similarly to blocking disk records. In BACKSPACE or FORESPACE operations, the correct tape positioning is accomplished by reading each block and determining the number of records within the block. All operations previously described for keyed or consecutive files apply except that <u>all</u> writes take place at the end of the file.

## ANS LABELED TAPE

An ANS labeled tape is given standard ANS format labels either through the ANS tape initialization processor (Label) or as the result of an operator key-in.

Some of the important advantages attendant to the use of ANS labeled tapes are:

- o Minimization of main memory space requirements for processing multiple volume sets.
- o Extended protection features relative to stated expiration dates.
- o Concatenated file processing which is particularly advantageous when a logical file has been generated in multiple sections.

The user should be aware, however, of the following restrictions on ANS labeled tape processing:

- o Tape cataloging is not available so Generation Data Groups are not applicable.
- o Blocking and deblocking are the responsibility of the user. (Standard COBOL library routines are available.)
- o Multiple tape sets are processed by serial number only.

## EXCLUSIVE USE OF TAPE FILES

Single-File Tapes. Once a user has opened a file on a tape, no other user may access the tape until the original user closes and removes it.

Multifile Tapes. Once a user has opened a file on a multifile tape, no other user may access the tape until the original user has closed and removed the tape. If the REW option is specified, the tape is rewound. Otherwise, the tape remains at the current position and, if a DCB is opened using tape, one of two actions occurs:

1. On input or update, the tape is scanned forward for the desired file.
2. On output, the tape is positioned to the end of the current file and the new file is written at that position.

## DEVICE INPUT/OUTPUT

Devices used for I/O other than file or labeled tape basically divide into three classes:

1. Interactive Terminals

2. Batch Type Devices (local or remote)

3. Unformatted Devices

A DCB used for I/O by a user program can become connected to these types of devices by assigning (ASSIGN, SET, M:OPEN) a DCB to a logical device or a physical device. If an assignment is made to a physical device, only the device type information is used. The actual device used is determined by the operating system. A logical device assignment can be to an operational label or a logical device stream. Operational labels are a set of convenient default assignments which are set up in a system basis with one set of assignments for on-line jobs and one set for batch jobs. A logical device stream is an input or output stream which can be connected to batch type devices with the LDEV command.

## INTERACTIVE TERMINALS

Interactive terminals fall into two classes, master and slave, with a great deal of commonality between classes. Regardless of type, terminals are usable as I/O devices with a great deal of flexibility. Terminals can be operated in echoplex mode or with local printing. Sophisticated editing and tabbing features are provided on input. When operating in echoplex mode, type ahead is allowed with proper sequencing of input and output guaranteed. Tabbing, line breakup to fit device, pagination, and page headers are provided on output. A wide variety of timing algorithms and code sets are provided to fit the idiosyncracies of specific terminals.

Slave terminals are acquired by programs as a contention resource (not pre-allocated) if not already acquired or logged on as a master terminal. The terminal may then be used as an I/O device by the program until released. Slave terminals are completely under control of the program.

Master terminals are associated on a one-to-one basis with on-line jobs. In addition to being an available I/O device, the terminal is the control device for the on-line job. TEL reads commands from the terminal. Break can be used as a program interrupt and Control Y causes an escape to the associated command processor (usually TEL). Thus the master terminal is in control of the program. A terminal becomes a master simply by logging on.

## BATCH TYPE DEVICES

All I/O to batch type devices (card readers, card punches, line printers, plotters) is staged via symbiont files whether the device is local or remote. This simply means that the entire input job from a card reader is read onto disk before processing and all of the job's output to these devices is stored on disk, not being output until the job is complete.

The processing of these devices symbiotically provides the following benefits:

- Disconnects program execution from I/O devices
- Smooths peaks and valleys in I/O demand
- Allows multiple programs to output to the same device simultaneously
- Allows grouping of output by form type
- Allows a program to generate several 'streams' of output to one device
- Allows several copies of output to be produced
- Enables on-line use of batch peripherals
- Allows natural submission of jobs from on-line terminals
- Allows pre-scanning of job requirements for efficient resource allocation in batch scheduling

## UNFORMATTED DEVICES

An unformatted device (primarily free-form tape) is handled as a resource which must be pre-allocated (contended for if on-line). When the device is allocated to the user he is responsible for the data read or written to the device. No blocking or formatting services are provided.

## SPECIAL FEATURES OF THE I/O SYSTEM

In addition to the facilities of the I/O system as outlined above, there are a number of special features which enhance the ease of use of CP-V.

- File Extension - All system DCBs (M:operational label) will be provided file extension by the system. Thus the first use of an output DCB within a job will create a new file and subsequent uses will add to the end of the file without explicit action on the part of the using program.
- JOB files which have :: as the second two bytes of the name will have the job id substituted, thus automatically preventing contention with another job in the same account.
- A procedure is available for a program to declare a file temporary (which causes it to be deleted at end of the job).
- Procedures are available to determine the next account in the account directory and the next file in a file directory.
- Assignments to the M:SI (source input) DCB by on-line jobs are extant for one job step only.
- A procedure (M:MOVE) is available to repetitively read and write records between two DCBs until an abnormal condition is reached.

## RELATED PROCESSORS

In addition to the use of files for program I/O and in conjunction with the use of various language and application programs, two processors are supplied with CP-V to be particularly useful in the manipulation of files.

## PERIPHERAL CONVERSION LANGUAGE

The Peripheral Conversion Language (PCL) is a utility processor designed for working with files in a batch or time-sharing environment. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, and RAD or disk pack storage.

The command language provides for single or multiple file transfers with options for selection, sequencing, formatting, and conversion of data records. Additional file maintenance and utility commands are provided.

Some examples of PCL commands are given below. All of the PCL commands are summarized in Table 5-1.

1. COPY LT#83/TAXFILE TO LP(K)

   Copies the file called TAXFILE which is stored on labeled tape 83 to the line printer, printing the key (K) associated with each record.

2. LIST

   Lists the names of all files in the user's file directory (i.e., all of the user's files which are stored on the system disk).

3. COPYALL TO DP#A23

   Copies all files in the user's file directory to disk pack A23

4. WEOF

   Writes an end-of-file on the user's current output device.

5. REW LT#1234

   Rewinds labeled tape 1234.

Table 5-1. PCL Command Summary

| Command | Function |
|---|---|
| COPY | Copies file(s) between devices or between public storage and devices. |
| COPYALL | Copies files from RAD, labeled tape, or disk pack to any output device. |
| COPYSTD | Copies a control file and all files named within the file. |
| DELETE | Deletes specified file(s). |
| DELETEALL | Deletes all files or a specified range of files in the job's account. |
| END | Returns control to the monitor. |
| LIST | Lists files names and, optionally, attributes from the account directory, tape, or disk pack. |
| REMOVE | Removes a magnetic tape or disk pack. |
| REVIEW | Lists files in the job's account and waits for a user response after listing each file name to allow the option of deleting the file. |
| REW | Rewinds a tape reel. |
| SPE LT | Spaces to the end of the last file on labeled tape. |
| SPF FT | Positions free form tape forward or backward a designated number of files. |
| TABS | Set tab values for tab expansion. |
| WEOF | Writes an end-of-file on the current output device. |

## EDIT PROCESSOR

Edit is a line-at-a-time context editor for creation, modification, and manipulation of files of EBCDIC text. It is only available to time-sharing users.

All Edit data is stored on disk in a keyed file structure of sequence-numbered variable-length records, which permits Edit to directly access each line or record of data. Edit functions are controlled via single-line commands from the user. The command language provides for the following:

1. Creating a sequenced EBCDIC coded text file.
2. Inserting, reordering, and replacing lines or groups of lines of text.
3. Selective printing and renumbering.
4. Reordering groups of records within a file.
5. Merging part of one file into another.
6. Context editing operations that allow matching, moving and substituting character strings within a specified range of text lines.
7. Maintaining files (allowing the user to build, copy, and delete whole files of text lines).

A user may edit files under his own account (i.e., the one under which he logged on) or under accounts to which he has been granted write access by the file creator. He may copy his own files or those to which he has read access. Under the rules of CP-V file access, a file may not be created (i.e., built or copied to) under an account number different than that used for log-on unless a very high privilege level is associated with the log-on account.

An example of the use of the Edit processor to create a file is given in Figure 5-3. (Output from CP-V is underscored. User input is not underscored.) All of the Edit commands are summarized in Table 5-2.

```
!BUILD PRIME
```

The user wants to create a file called PRIME. (Edit is called implicitly by the BUILD command.)

```
 1.000  10  REM    GENERATE PRIMES GR THAN #
```

Edit prompts for input by printing 1.000. The user types the first line, then types lines 2-10 in response to more prompts by Edit.

```
 2.000  20  P=1
 3.000  30  P=P+4,S=0
 4.000  40  FOR I = 5 TO SQR(P) + 1 STEP 2
 5.000  50   Q=INT(P/I)
 6.000  60   IF Q*I=P THEN 80
 7.000  70  PRINT P''TAB(0)
 8.000  80  IF S=1 THEN 30
 9.000  90  S=1, P=P+2
10.000  100  GOTO 40
11.000
```

The user types a carriage return immediately following the prompt for line 11.000 to indicate end-of-file, that is, that the last line of the file has been entered.

Figure 5-3. Sample Edit Session

Table 5-2. Edit Command Summary

| Command | Function |
|---|---|
| BP | Sets the blank preservation mode. When "on", all strings of blanks are preserved during intrarecord operations. When "off", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off". |
| BUILD | Enables the user to create a new file. |
| CM | Causes Edit to insert commentary into specified columns of each successive record beginning at a specified record. |
| COPY | Copies one file to another file. |

Table 5-2. Edit Command Summary (cont.)

| Command | Function |
|---|---|
| CR | Controls the inclusion of the carriage return character (X'15') at the end of each record in the user's output file. |
| CT | Causes Edit to type the record up to a specified column and then to insert commentary (given by the user) into specified columns of each successive record beginning at the specified record. |
| D | Locates a given occurrence of an indicated string, between columns specified by an SE, SS, or ST command, and deletes it. |
| DE | Deletes all records whose sequence numbers lie in a specified range. |
| DELETE | Deletes the file specified by fid from the log-on account. |
| E | Starts at a column occupied by the first character of a given occurrence of a specified string or column and overwrites with another string. |
| EDIT | Opens a file to be edited and enters the record editing mode. |
| END | Closes all active files and returns control to the terminal executive language (TEL). This command is equivalent to the X command. |
| F | Starts after the last character of a given occurrence of a specified string or column and inserts another string, pushing everything from this column right as required to make room. |
| FD | Searches for the specified string between specified columns in a specified range of records. If the string is found, the record containing it is deleted from the file. |

Table 5-2. Edit Command Summary (cont.)

| Command | Function |
|---|---|
| FS | Searches for the specified string between specified columns in a specified range of records. Each time the string is found, the sequence number of the record is printed. |
| FT | Searches for the specified string between specified columns in a specified range of records. Each time the string is found, the sequence number and the contents of the record are printed. |
| IN | Inserts new records into a file starting at a specified record. Edit prompts the user with the sequence number of each record to be inserted. |
| IS | Inserts new records into a file starting at a specified record. Edit does not prompt with sequence numbers of the records to be inserted. |
| JU | Causes the SS or ST command to jump to the specified record and then continues stepping from that point. |
| L | Shifts portions of the record left the number of positions indicated. |
| MD | Moves records within a file from one range to another range. The original records are deleted. |
| MERGE | Merges records from one file into another file. |
| MK | MK is identical to MD except that the original records are not deleted as they are moved. |
| NO | Specifies that no editing is to be performed on the current active line. |

Table 5-2. Edit Command Summary (cont.)

| Command | Function |
|---|---|
| O | Starts at the column occupied by the first character of a given occurrence of a specified string or column and overwrites with another string. |
| P | Starts before the first character of a given occurrence of a specified string or column and inserts another string, pushing characters of the first string to the right as required to make room. |
| R | Shifts portions of the record right the number of positions indicated. |
| RF | Causes the current setting of the blank preservation mode ("on" or "off") to be reversed temporarily (for the current line only). |
| RN | Renumbers a specified record. |
| RP | Sets the record size preservation mode. When ON, the size of the edited records is not changed. Trailing blanks are not deleted. When OFF, records are shortened or lengthened as necessary by the editing process. Trailing blanks are deleted. The default mode is OFF. |
| S | Locates a specified string between columns specified by an SE, SS, or ST command and replaces it with another string. |
| SE | Causes Edit to accept successive lines of intrarecord commands to be applied to records beginning at a specified record. |
| SS | Causes Edit to start at a specified record and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. |
| ST | Causes Edit to start at a specified record and proceed to each record in succession, |

Table 5-2. Edit Command Summary (cont.)

| Command | Function |
|---|---|
| ST (cont.) | accepting one line of intrarecord commands to update the current record. The sequence number and contents of each record are typed prior to accepting a command. |
| TA | Causes Edit to set or reset the terminal tab stops to settings appropriate for a specified language processor. |
| TC | Types the sequence numbers and the contents of specified columns of one or more records beginning at a specified record. Any nonblank strings within the columns typed are shifted to the left to compress each blank string to a single blank. |
| TS | Types the contents of the record currently open for editing under control of an SE, SS, or ST command or types the contents of specified columns of one or more records beginning at a specified record. |
| TX | Types the sequence number and contents of those records within the edit range (set by SE, SS, or ST commands) which have been changed by the preceding intrarecord command(s). |
| TY | Types the sequence number and contents of the record currently open for editing under control of an SE, SS, or ST command or types the sequence numbers and the contents of specified columns of one or more records beginning at a specified record. |
| X | Closes all active files and returns control to the terminal executive language (TEL). This command is equivalent to the END command. |

# CHAPTER 6. USER PROGRAM SERVICES

## SYSTEM PROCEDURES

System procedures (procs) are the single communication mechanism between user programs and the CP-V system during program execution. They form the command language by which program requests are communicated to the system. Each procedure consists of a CAL instruction and an associated parameter list describing the details of the request. A set of Meta-Symbol procedures (macros) may be used to generate the correct CAL instruction and parameter list.

Procedures have several properties guaranteed by the system. All registers remain undisturbed by the execution of a proc (except for certain procs which return results in designated registers). The parameter lists describing the requests are generated out of code sequence with the CAL. (Instead, they are placed together in a separately protected section of the program.) All parameters are checked for validity by the system before use. CP-V will not accept or use bad parameters. With certain obvious exceptions, all procedures act identically regardless of whether the program is executing in a batch or time-sharing mode. After normal execution, the return from a procedure is to the point just following the CAL instruction which called the procedure.

The CP-V system procedures are usually spoken of by their formal procedure name. Brief descriptions of all the procedures are given in Table 6-1. The procedures are listed in alphabetical order.

Table 6-1. System Procedures

| Procedure | Function |
|---|---|
| M:AND | Causes a specified test to be made at a specified location. Only if the condition is true and the specified test identifier is set does it remain set; otherwise, it is reset or remains reset (see M:SNAPC procedure). |
| M:BLIST | Converts a virtual command list to a physical command list. |
| M:BUFSTAT | Checks the buffer status of a slave character oriented time-sharing line. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:CHECK | Checks type of I/O completion. |
| M:CHECKECB | Checks for the completion of an event or a set of events. |
| M:CHKINT | Checks an interrupt status. This procedure is for real-time processing. |
| M:CLOCK | Enters a program at a particular location after a specified period of clock time has passed. |
| M:CLOSE | Terminates all I/O associated through a given Data Control Block (DCB). This can cause a variety of actions depending on assignment. |
| M:COC | Sends a character to a character oriented time-sharing terminal. This procedure is for real-time processing. |
| M:CONNECT | Connects a real-time program address to an interrupt. |
| M:COUNT | Specifies the range and the steps within the range where a specified test identifier is set (see M:SNAPC procedure). |
| M:CT | Changes terminal type, activation set, and other terminal attributes; examines current status; controls terminal coupling. |
| M:CVM | Changes the Virtual Map by placing a given real page number in a designated virtual page map location. |
| M:CVOL | Causes the control program to advance to the next volume of a data set before the physical end of the current volume is detected. |
| M:DCB | Defines and generates a Data Control Block. This procedure does not create any code for execution. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:DCLOSE | Closes a DCB that was opened to a device in the diagnostic mode. |
| M:DDCB | Creates a diagnostic DCB. This procedure does not create any code for execution. |
| M:DELREC | Specifies that a data record is to be deleted from the file. |
| M:DEQ | Dequeues resources which were queued with M:ENQ. |
| M:DEVICE | Allows the user to set special device parameters or invoke special procedures. There are 14 different procedures invoked by options of this procedure. |
| M:DISCONNECT | Disconnects an interrupt from a program address. |
| M:DISPLAY | Reports system load parameters. |
| M:DOPEN | Opens a DCB to a device in the diagnostic mode. |
| M:DMOD# | Obtains a device model number. |
| M:ENQ | Allows two or more programs to coordinate activities by forming common queues. This is often used for coordinating access to shared file records. |
| M:ERR | Causes a job exit with an error. A batch program continues as specified by the next STEP command. Any on-line program continues as directed by the user. |
| M:EXCP | Executes a channel program. This procedure is for real-time processing. |
| M:EXIT | Returns control to the monitor which then continues with the next control command from batch stream or on-line user. |
| M:EXU | Requests that the monitor execute a privileged instruction for the user. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:FP | Frees pages of main storage owned by a given program. |
| M:FCP | Frees common pages. |
| M:FVP | Frees virtual pages. |
| M:GCP | Gets common pages. |
| M:GDDL | Gets dynamic data limits on main storage. |
| M:GDG | Gets a disk granule for a real-time program. |
| M:GET | Retrieves system parameters from the Job Information Table. |
| M:GETID | Gets a Transaction Processing Queue identifier. |
| M:GETLINE | Gets a slave character oriented time-sharing line. |
| M:GJOB | Initiates a ghost job. |
| M:GJOBCON | Associates a real-time ghost program with an interrupt. |
| M:GL | Gets common limits on main storage. |
| M:GP | Allocates pages of main storage to the requesting task. |
| M:GVP | Gets virtual pages. |
| M:HOLD | Holds a real-time program in main memory (i.e., prevents swapping). |
| M:IF | Causes a specified test to be made at a specified location. Only if the specified test condition is true is the test identifier set; otherwise, it is reset or remains reset (see the M:SNAPC procedure). |
| M:INHIBIT | Inhibits a real-time interrupt. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:INT | Requests program control at a particular location when a console interrupt occurs. |
| M:INTCON | Controls a real-time interrupt status. |
| M:INTRTN | Causes a return from a real-time interrupt processing routine. |
| M:INTSTAT | Querys the status of a real-time interrupt. |
| M:IOEX | Issues an I/O command list to a real-time device. |
| M:JOB | Inserts a file into the symbiont files (for batch execution usually) or deletes an existing symbiont file. |
| M:KEYIN | Writes the specified messages to the operator on the operator's console and returns the operator's reply to the program issuing the procedure. |
| M:LDEV | Attaches an information stream to a physical device (identified by a logical device name) and defines attributes of the physical device. |
| M:LDTRC | Loads the specified program load module (if a shared copy is not available in memory), deletes the calling module, and transfers control to the loaded load module. |
| M:LINK | Loads the specified program load module (if a shared copy is not available already in memory) and transfers to it after filing a copy of the calling program for possible return. That is, it uses the called module like a subroutine. COBOL programs call SORT this way. |
| M:LOCK | Either locks a diagnostic program in main memory or allows normal swapping to resume for it. |
| M:MAP | Converts a virtual address to a physical main memory address. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:MASTER | Requests that the program be allowed to operate in the master mode. |
| M:MDFLST | Modifies the terminal polling lists. |
| M:MERC | Allows the user to have the monitor process any system abnormal or error code, overriding an ABN or ERR exit. |
| M:MESSAGE | Writes the specified message on the operator console. |
| M:MOVE | Copies an entire file, record by record. |
| M:NEWQ | Requests I/O to be performed without a DCB and without a user-built channel program. This procedure is for real-time processing. |
| M:OPEN | Causes the specified file associated with the specified DCB to be opened for use. This procedure causes a wide variety of actions depending on the DCB assignment. |
| M:OR | Causes a specified test to be made at a specified location (if a specified test identifier is reset). If the condition is true, the specified test identifier is set; otherwise, it remains unchanged (see the M:SNAPC procedure). |
| M:PC | Sets a prompt character. |
| M:PT | Allows the user to generate FPTs in either protected or unprotected storage. (This is an assembler directive, not a CAL.) |
| M:PFIL | Causes the specified tape to be positioned past the number of file marks specified and in the direction specified. |
| M:PRECORD | Causes the tape specified by the DCB to be positioned in the direction specified by the specified number of records. |
| M:PRINT | Writes the specified message on the listing log output media. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:PURGE | Purges I/O on a character oriented time-sharing line. |
| M:QFI | Queues a real-time program pending an interrupt. |
| M:QUEUE | Enters and retrieves transactions in the Transaction Processing Queue. |
| M:RAMR | Reads the assign/merge record. |
| M:RDG | Releases disk granule for a real-time program. |
| M:READ | Causes a logical data record to be read into the location specified by the user from a device or file. |
| M:REW | Rewinds a tape specified by the DCB. |
| M:RLSLINE | Releases a slave character oriented time-sharing line. |
| M:RRESOURCE | Releases resources (e.g., disks, tapes). |
| M:RUE | Reports an event to the CP-V scheduler. This procedure is for real-time processing. |
| M:SAVE | Saves system parameters in the Job Information Table. |
| M:SEGLD | Loads a specified program overlay segment into memory. |
| M:SETDCB | Sets error or abnormal addresses in a specified DCB. |
| M:SLAVE | Allows any master mode program to return to the slave mode. |
| M:SMPRT | Sets memory protection. |
| M:SNAP | Causes a snapshot of the registers and memory specified to be printed. |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:SNAPC | Causes a snapshot of the registers and memory specified to be printed if the specified test identifier is set. Whether the test identifier is set or not is dependent on the M:IF, M:AND, M:OR, and M:COUNT procedures. |
| M:STARTIO | Returns a real-time device to the system. |
| M:STIMER | Sets the interval timer with a specified interval of program execution time. |
| M:STOPIO | Stops I/O on the designated real-time device. |
| M:STRAP | Simulates a trap. |
| M:SUPCLS | Terminates current symbiont program output and initiates output to the designated device. |
| M:SYS | Allows a program to use privileged services. |
| M:TFILE | Causes a specified DCB to be closed, on return to the user's program, and the associated file to be registered as a scratch file for deletion at the end of the job. |
| M:TIME | Communicates the time of day and the current date to the executing program. |
| M:TRAP | Allows a program to take control at a particular location on occurence of a machine trap. |
| M:TRTN | Restores control to the executing program from a trap or timer routine. |
| M:TRUNC | Causes the blocking buffer reserved for a specified DCB to be released (written if necessary). |
| M:TTIMER | Gives the time remaining in the interval that was previously set by an M:STIMER |

Table 6-1. System Procedures (cont.)

| Procedure | Function |
|---|---|
| M:TTIMER (cont.) | procedure and optionally cancels the interval in effect. |
| M:TYPE | Writes the specified message to the operator on the operator's console. |
| M:WAIT | Suspends the program for a certain period of time. |
| M:WAMR | Writes the assign/merge record. |
| M:WEOF | Writes an end-of-file mark on an unlabeled tape specified by the DCB. |
| M:WRITE | Causes the contents of a specified buffer to be transmitted to an output device or file as a logical record. |
| M:XCON | Allows a program to regain control after program termination. |
| M:XXX | Causes the monitor to terminate the job and not honor any further commands (if the program is running in the batch mode). |

## MONITOR ERROR MESSAGES

Each monitor error message is associated with a two digit code and a two digit subcode. The code defines the general problem and the subcode categorizes the problem more specifically. There are two types of monitor codes - abnormal codes and error codes. Abnormal codes are those with numbers less than 40 and error codes are those with numbers of 40 or greater. This reflects the fact that error conditions are those considered to be of greater severity than abnormal conditions. In fact, if an abnormal condition occurs, the system will by default attempt to proceed. If an error condition occurs, however, the system will by default abort the job. In both cases, the user may override the system default and specify that the condition is to be handled by user-developed code. If the user asks for control, the error or abnormal code is returned in a specified location and no error message is printed. Otherwise, the monitor prints the message corresponding to the code and takes the default action.

# CHAPTER 7. DEBUGGING FACILITIES

## TIME-SHARING MODE DEBUGGING

Errors that occur in a user program which is running in the time-sharing mode are reported directly at the terminal. This allows the user to take immediate action to correct the error. Thus, an important advantage of the time-sharing mode of operation is that it allows the user to debug programs (i.e., locate and correct errors) dynamically. Due to the intermode compatibility, programs created for the other modes of operation may be debugged in the time-sharing mode.

There are three processors which were designed to facilitate on-line debugging. They are:

- Delta - for debugging programs written in assembly language. (However, Delta works at the machine language level and may be used to debug any program.)

- FORTRAN Debug Package (FDP) - for debugging programs written in FORTRAN.

- COBOL On-line Debugger - for debugging programs written in COBOL.

The general facilities provided by these debugging processors allow the user to:

1. Examine, insert, and modify such program elements as instructions and data.

2. Trace the flow of program execution.

3. Control the flow of program execution.

The Delta commands are summarized in Table 7-1, the FDP commands are summarized in Table 7-2, and the COBOL On-line Debugger commands are summarized in Table 7-3.

Table 7-1. Delta Command Summary

| Type of Commands | Functions |
|---|---|
| Expression Evaluation | Evaluates expressions consisting of program symbols, special symbols, assembly language mnemonics, and certain operators. |
| Memory Cell Opening and Display | Displays the contents of a memory cell or a series of cells and opens it (or them - one at a time) for modification as described next. |
| Memory Modification | Assembles an expression and stores it in the currently open cell and closes the cell. Commands exist that allow a user to conveniently modify a series of cells in this manner. |
| Symbol Table Control | Selects a particular symbol table for use in debugging; loads the global symbol table; displays undefined symbols; allows the user to define new symbols; and removes selected symbols. |
| Execution Control | Begins execution at a specified location or continues execution from the point at which execution was interrupted. |
| Snap Dump Control (Executive Delta only) | Produces hexadecimal dumps on the line printer or to disk. |
| Overlaid Program Control | Allows Delta to break when an overlay is loaded. |
| Breakpoints | Requests that Delta break when a specified instruction is encountered, when a specified memory cell is modified, or when a branch instruction is executed (and branches). Several instruction break points may be set. Optionally, the user may |

Table 7-1. Delta Command Summary (cont.)

| Type of Commands | Function |
|---|---|
| Breakpoint (cont.) | trace execution by displaying information at designated points in the program. |
| Memory Search and Modification | Searches the program and data for a specified element. Memory is only searched between user-specified bounds. |
| Memory Clearing and Setting | Zeros memory between specified locations or sets memory between specified locations to a specified value. |
| Display Modes | Allows the user to control the mode of displays. For example, the user can set the display mode for memory addresses to symbol plus relative hexadecimal offset. |
| Printer Output | Prints the contents of memory between specified locations. The output can go to the line printer or the user's terminal. |
| Disk Dumps (Executive Delta only) | Prints the contents of the disk from a specified area. |
| Prompt Character Changing | Changes the prompt character for Delta to a user-specified character. |
| Program Termination | Disassociates Delta and exits to TEL. |

Table 7-2. FORTRAN Debug Package (FDP) Commands

| Command | Function |
|---|---|
| GO | Start or resume execution. |
| single break | Interrupt execution at next FORTRAN statement. |
| double break | Interrupt execution immediately. |

Table 7-2. FORTRAN Debug Package (FDP) Commands (cont.)

| Command | Function |
|---|---|
| QUIT | Quit the debugging run. |
| stepping | Resume execution and step to a subsequent FORTRAN statement (backtracking mode not set). |
| backtracking | Display flow history transactions (backtracking mode set, see the HISTORY command). |
| RESTART | Restart at the beginning of the main program (no data or files are initiated). |
| REWIND | Rewind files. |
| ABORT LEVEL | Set the abort level. |
| LOC | Display the location of a FORTRAN statement. |
| PRINT,<br>OUTPUT, or ? | Display the value of variables, arguments, or positions. The value of a position is the source line number for that position. The values of arguments (may only be displayed during calling sequence breaks) and variables depend on the type of the item, but hexadecimal or string values are optional. |
| value change | Change the values of variables. During calling sequence breaks, the values of arguments (actually, the variables they represent) may be changed. |
| GOTO | Branch to a specified position. |
| FLOW | Turn on flow tracing mode. |
| NOFLOW | Turn off flow tracing mode. |
| HISTORY | In on-line runs, set backtracking mode and, possibly, display the most recent flow history transactions (in reverse order). In batch runs, display the most recent flow history transactions (in flow order). |

Table 7-2. FORTRAN Debug Package (FDP) Commands (cont.)

| Command | Function |
|---|---|
| RESET HISTORY | Erase the current flow history transactions to avoid duplicate didsplays. |
| USE FILE | Send run-time debug output to a specified file for later analysis. |
| USE ME | Send run-time debug output to the terminal. |
| KILL | Revoke debug commands. |
| SKIP | Skip a FORTRAN statement or a series of FORTRAN statements. |
| AT | Set statement break. |
| ON | Set data break. |
| ON CALL | Set specific calling sequence break. |
| ON CALLS | Set to break on all calling sequences not covered by a specific ON CALL. |

Table 7-3. COBOL On-line Debugger Command Summary

| Command | Function |
|---|---|
| AT | Establishes a breakpoint at a specific location in the program. |
| WHEN | Establishes a breakpoint that is effective whenever the contents of an identifier are changed. |
| STOP | Halts execution of the program at a breakpoint established by an AT or WHEN command. |
| IF | Provides conditional execution of debugger commands at a breakpoint established by an AT or WHEN command. |
| OFF | Removes a breakpoint that was established by an AT command. |

Table 7-3. COBOL On-line Debugger Command Summary (cont.)

| Command | Function |
|---|---|
| OFFWN | Removes a breakpoint that was established by a WHEN command. |
| OFFS | Removes all breakpoints that were established with an AT command and have a statement-id as their location. |
| OFFP | Removes all breakpoints that were established with an AT command and have a procedure-name as their location. |
| LISTBRKS | Lists all the breakpoints that are currently established. |
| BREAK (key) | Establishes a temporary breakpoint before the execution of the next COBOL statement. |
| RUN | Removes all breakpoints established by AT or WHEN commands and continues execution of the program. |
| NEXT | Executes a single statement of the program. |
| NEXTP | Resumes execution of the program until the next procedure-name is encountered. |
| GO | Begins execution of the program at a specified location or at the current statement of the program. |
| PRINT | Causes the contents of an identifier to be displayed on the terminal in the natural mode of the identifier. |
| PRINTX | Displays the contents of an identifier in hexadecimal. |
| SET | Places a specified value in an identifier. |
| EQUATE | Enters an abbreviation for an identifier into the abbreviation table. |
| DROP | Removes an abbreviation from the abbreviation table. |

Table 7-3. COBOL On-line Debugger Command Summary (cont.)

| Command | Function |
|---|---|
| DUMP | Provides a hexadecimal dump in the standard format for post mortem dumps. |
| STRACE | Controls statement execution trace mode. |
| SLIST | Displays the history of statement execution. |
| PTRACE | Controls the procedure-name execution trace mode. When the procedure trace mode is on, each procedure-name is printed out prior to execution of the procedure. |
| PLIST | Lists the contents of the procedure-name execution history table. |
| SOURCE | Prints one or more source lines on the programmer's terminal. |
| REPLACE | Allows the programmer to replace an existing source statement. |
| INSERT | Allows the programmer to add new lines to the source file. |
| DELETE | Deletes a line from the source file. |
| SETFILES | Provides the programmer with a quick and easy method for ensuring that all his DCBs are correctly assigned. |
| LISTFILE | Displays the status of an FD in the program. |
| HELP | Lists the debugger commands, with a short description of each. |
| QUALIFY | Changes the default value of the program-id, the name of the source-file to be used in source manipulation commands, or the debug file to be used. |
| END | Terminates the debugging session and returns control to the monitor. |

## BATCH PROCESSING MODE DEBUGGING

Batch program errors are reported via either a default mechanism or through explicit dump and snap commands supplied by the user in his JCL deck or internally within his program.

Errors occurring during the execution of a batch user program are reported to the user via the error codes and subcodes as described in Chapter 6. If the user does not choose to handle these errors himself (i.e., does not use the debug commands), the monitor aborts the job and interprets the codes for him by accessing the error message file for an appropriate message. This message is printed together with the location of the error, the PSD, the general registers, and, if the error is DCB-related, the contents of the DCB. For example:

```
    4000 CAN'T READ AN OUTPUT FILE

    AT C065

    ON DCB M:EO

    WHICH CONTAINS

    (contents of DCB)

    USER's PROGRAM STATUS DOUBLEWORD

    (contents of PSD)

    USER's GENERAL REGISTERS

    (contents of registers)
```

The memory dumps performed by debug commands may be either conditional (dependent on whether errors occurred during program execution) or unconditional. All dumps are taken before the DCBs are closed.

## POSTMORTEM DUMPS

A postmortem dump control command requests the monitor to dump a selected area of memory. Such a dump is termed "postmortem" because it is performed after the program has been executed or terminated due to error (i.e., "errored"). If an error is detected during program execution, the monitor lists an appropriate error message in addition to listing the dump output.

Postmortem dumps are requested by various forms of the PMD system control command. The two forms PMD and PMDE cause various portions of the user's environment to be dumped if an error occurs. The form PMDI causes a dump to occur whether or not any errors have been detected.

## SNAPSHOT DUMPS

A memory snapshot dump provides an instantaneous "picture" of program conditions existing at a particular point in time during program execution. Such a dump can be obtained just prior to the execution of any specified instruction in a user's program. Six control commands and six equivalent procedures are provided for specifying the circumstances that will produce a snapshot dump and the portion of memory that the dump will include. These are summarized in Table 7-4.

Table 7-4. Snapshot Dump Control Commands and Procedures

| Control Command | Procedure | Function |
|---|---|---|
| SNAP | M:SNAP | Requests the monitor to take an unconditional memory snapshot. |
| SNAPC | M:SNAPC | Requests the monitor to take a conditional memory snapshot. |
| IF | M:IF | May be used in conjunction with conditional snapshots (SNAPC, M:SNAPC). It requests the monitor to make a specified test at a designated location and, if the condition is found to be true, to set the flag bit associated with the conditional snapshot. |
| AND | M:AND | May be used in conjunction with conditional snapshots (SNAPC, M:SNAPC). It requests the monitor to make a location, but only if the flag bit for the associated snapshot is in |

Table 7-4. Snapshot Dump Control Commands and Procedures (cont.)

| Control Command | Procedure | Function |
|---|---|---|
| AND (cont.) | M:AND | the set state when the test is to be made. |
| OR | M:OR | May be used in conjunction with conditional snapshots (SNAPC, M:SNAPC). It requests the monitor to make a location, but only if the flag bit for the associated snapshot is in the reset state when the test is to be made. |
| COUNT | M:COUNT | Allows the user to specify an iteration range (and steps within that range) in which a flag for a snapshot dump is to be set if a user-specified count is within a user-specified range. |

## SYSTEM DEBUGGING

Extensive facilities are provided for debugging the CP-V system itself. Facilities of specific importance are described below.

## AUTOMATIC CRASH ANALYSIS

Explicit internal tests detect most software and some hardware problems and activate the system recovery routines. A dump taken at this time is submitted to a ghost job (ANLZ) which provides an analysis of the problem, including the immediate symptoms and formatted presentations of system tables and job-dependent information. About 20 specially formatted tables are presented in terms of well-known system symbols, permitting fast and accurate isolation of the problem either by home office experts or by field analysts. The analysis and repair of problems is thus especially prompt.

## REMOTE SYSTEM ANALYSIS

The same routines used for automatic analysis may be used during system operation from any remote or local terminal to isolate system problems in a crash dump on file or in the running system itself. Further, the language Delta may be used to examine and repair the running monitor, again from a remote terminal. This facility has proven especially valuable as a fast, money-saving aid to solving problems at CP-V installations; it allows several home office experts to combine their various talents to solve a problem within hours rather than making long, expensive trips to the customer's site. They can log onto the customer's system, gather information, analyze it, and dynamically apply the correcting patch.

## EXECUTIVE DELTA

For especially difficult problems which require hands-on control, the executive debugger, XDELTA, provides for complete total control of the operating system. Breakpoints may be set to stop the system for examination at crucial points using the symbols of the system for accurate, easy-to-read displays.

## SYMBOLIC PATCHING

XDELTA is also used to patch the monitor at boot time. The same symbolic patch format is used for both debugging and for patching. Patches are generally relocatable so that patch decks may be applied without change to all system regardless of the SYSGEN configuration.

## GENMD PATCHING

The GENMD processor permits on-line, batch, and ghost users to make permanent modifications to existing load modules, thereby reducing the number of compilations required to debug a program. GENMD patches are used to modify nonresident elements of the system.

## BOOT-UNDER-THE FILES

Both GENMD patches and XDELTA patches may be applied to the system at boot-time. CP-V also provides for rereading the patch deck without disturbing the permanent files of the system and its users, which avoids the necessity of saving and restoring an extensive file system in order to apply critically needed patches.

# CHAPTER 8. TIME-SHARING

## INTRODUCTION

For those activities best suited to an interactive environment, CP-V provides time-sharing service for remotely connected terminals. A variety of language and utility processors are provided to aid the user in accomplishing:

- Program development
- Program compilation
- Program execution
- Program debugging
- File maintenance
- Text creation and editing

Programs to be executed in the other modes of operation may be completely or partially developed in the time-sharing mode.

More than 128 terminals may be connected simultaneously depending on hardware configuration, activity at terminals, and desired response time. The following types of terminals may be used with CP-V:

Xerox Model 7015 Keyboard/Printer.

Xerox Model 3010 Keyboard/Printer.

Teletype Models 33, 35, 37, and 38.

IBM 2741 Terminals.

Tektronix Models 4010 and 4013.

Datapoint 3300.

Any terminal compatible with any of the above.

CP-V allows terminals to be permanently connected or to be temporarily connected via dial-up communication circuits. Local terminals may be connected permanently through Xerox-supplied line interfaces. Remote terminals requiring permanent connections are provided through leased communication circuits.

Regardless of how a terminal is physically connected to CP-V, terminal protocol is the same. After connection has been established, users identify themselves by entering their account, their name, and if required, a password. If the identification is valid and consistent with information maintained by the monitor, the user's on-line session is initiated and the system prompts the user for commands. If the identification is invalid, CP-V sends an error message and requests the user to resupply "log-on" data.

An on-line session is terminated by entering a simple "log-off" command or by hanging up the telephone connnection. CP-V then transmits selected accounting information and offers the user the opportunity to log on again. Thus, separate accounting for separate functions may be achieved by a change of account number and/or name.

Concepts that are relevant only to the time-sharing mode of operation are discussed in the remainder of this chapter.

## TERMINAL EXECUTIVE LANGUAGE

The Terminal Executive Language (TEL) is the principal terminal language for the system. Most activities associated with COBOL, FORTRAN and assembly language programming can be carried out via TEL commands. A summary of the TEL commands is given in Table 8-1.

Table 8-1. TEL Command Summary

| Command | Description |
|---|---|
| ANSF | Compiles an ANS FORTRAN source program. |
| BACKUP | Saves the specified file on a system tape. In case of a crash in which files are lost, files on the tape will be restored. |
| BATCH | Enters the specified file(s) in the batch job stream. |

Table 8-1. TEL Command Summary (cont.)

| Command | Description |
|---|---|
| BUILD | Allows a new file to be created from the terminal using the Edit processor. |
| BYE | Disconnects the terminal from the system and provides an accounting summary. This command is equivalent to the OFF command. |
| CANCEL | Cancels previously submitted batch jobs. |
| COBOL | Compiles an ANS COBOL source program. |
| COMMENT | Directs error commentary to the specified device or counteracts the preceding DONT COMMENT command. |
| COPY | Copies a file or device input to the specified file or device. |
| COUPLE | Allows other terminals to couple to this terminal. |
| COUPLE line number | Establishes a link between the user's terminal and the terminal specified by line number. |
| DECOUPLE | Releases the coupling between two terminals. |
| DELETE | Deletes the specified file(s). |
| DELTA | Calls the Delta debugging processor. |
| DISPLAY | Lists the current values of various system parameters. |
| DONT COMMENT | Stops error commentary output. |
| DONT COUPLE | Causes attempts to couple to the terminal to be rejected. |
| DONT LIST | Stops listing output. |
| DONT OUTPUT | Stops object output. |

Table 8-1. TEL Command Summary (cont.)

| Command | Description |
|---|---|
| DONT SEND | Disallows messages from the machine operator to the user's terminal. Global broadcasts are deferred until TEL is in control. Also disallows the MESSAGE command. |
| EDIT | Calls Edit to modify a file. |
| END | Terminates the current job step. This command is equivalent to the STOP and QUIT commands. |
| ERASE | Deletes the accumulated output for the line printer. |
| EXTEND | Sets the extended memory mode; i.e., appends the special processor area to the available user area. |
| FORT4 | Compiles an Extended FORTRAN IV source program. |
| GET | Restores the previously saved main memory image. This command is equivalent to the RESTORE command. |
| GO | Continues processing from the point of interruption. This command is equivalent to the CONTINUE and PROCEED commands. |
| JOB | Requests the status of jobs that were submitted to the batch queue via the Batch processor. |
| L | Lists file names and, optionally, attributes from the account directory, tape, or disk pack. |
| LDEV | Modifies a logical device definition; directs a stream of information. |
| LIST | Directs the listing output to the specified device, or counteracts the preceding DONT LIST command. |

Table 8-1. TEL Command Summary (cont.)

| Command | Description |
|---|---|
| LYNX | Forms the load modules as specified. |
| MESSAGE | Sends the specified message to the operator. |
| META | Assembles the specified Meta-Symbol source program. |
| OFF | Disconnects the terminal from the system and provides an accounting summary. This command is equivalent to the BYE command. |
| OUTPUT | Directs object output to the specified file, or counteracts the previous DONT OUTPUT command. |
| PAGE | Resets the terminal header page number to the value specified by n. |
| PASSWORD | Assigns, changes, or deletes a log-on password for the user. |
| PLATEN | Sets the value of the terminal platen width and/or page length or displays the terminal platen width and page length values. |
| PRINT | Sends accumulated symbiont output, such as output for the line printer or the card punch, to the output device. |
| PROCEED | Continues processing from the point of interruption. This command is equivalent to the GO and CONTINUE commands. |
| Processor Calls | These calls are entered while TEL is in control of the terminal. They turn over control of the terminal to the processor. Examples are:<br>APL BASIC PCL<br>FLAG LYNX |

Table 8-1. TEL Command Summary (cont.)

| Command | Description |
|---|---|
| program name | Initiates execution of the specified program. |
| QUIT | Terminates the current job step. This command is equivalent to the STOP and END commands. |
| RESET | Resets all DCBs back to their system default values. |
| RESTORE | Restores the previously saved main memory image. This command is equivalent to the GET command. |
| RUN | Loads the specified module and starts execution. |
| SAVE | Saves the current main memory image on the designated file. |
| SEND | Allows messages from the machine operator to be printed on the user's terminal. |
| SET | Assigns file or device to a DCB or sets DCB parameter. |
| SHOW | Displays information about currently logged-on user. |
| START | Loads a load module into main memory and starts execution of the program, either with or without an associated debugger. |
| STATUS | Displays the current accounting values. |
| STOP | Terminates the current job step. This command is equivalent to the END and QUIT commands. |
| SWITCH | Controls setting and resetting of the user's pseudo sense switches. With no parameters, the command displays the pseudo sense switch settings. |

Table 8-1. TEL Command Summary (cont.)

| Command | Description |
|---|---|
| TABS | Sets simulated tab stops for the terminal or displays the simulated tab stop settings. |
| TERMINAL | Sets the terminal type for proper I/O translations. |
| TERMINAL STATUS | Lists the terminal type and the current values of parameters associated with its operation. |
| TP | Logs off a time-sharing terminal and makes it available as a slave Transaction Processing terminal. |
| U | Causes the words UNDER DELTA to be inferred in the next command. |
| WHERE | Returns the line number of the specified user (if the user is logged on). |
| XEQ | Initiates processing of TEL commands from a command file. |

## SAMPLE TIME-SHARING SESSION

Figure 8-1 presents a sample time-sharing session. Output from CP-V is underscored. User input is not underscored.

```
XEROX CP-V AT YOUR SERVICE
ON AT 15:28 MAR 28, '76
LOGON PLEASE:  2232,HALL

ON AT 15:28  03/28/76 2232 B-7A

!BASIC
```

The user calls the BASIC processor, and begins to build a program, entering a BASIC statement in response to each prompt character.

Figure 8-1. Program Building, Editing, and Execution Using Basic (cont.)

```
>10 REM SAMPLE PROGRAM
>15 REM  $A IN STMT 20 IS A STRING VARIABLE
>20 $A = "COMPUTE ARCSINE OF X, IN DEGREES"
>30 PRINT $A
>40 FOR I -\= 1 TO 5
```

After typing the minus-sign (or dash) character by mistake - i.e., by forgetting to shift - he uses a Delete or Rubout character (echoed as \) to erase it and continues.

```
>50   INPUT X
>60   PRINT DEG (ASN(XX)) " = ARCSIN OF "X
>70 NEXT I
>80 END
>RUN
```

He enters the final statement (step 80) and then requests compilation and execution with the RUN command.

```
16:18    NOV 09 RUNIDAA...
COMPUTE ARCSINE OF X, IN DEGREES
?.001
 5.72958E-02 = ARCSIN OF 1.00000E-03
?.707
 44.9913 = ARCSIN OF .707000
?-0.707
-44.9913 = ARCSIN OF -.707000
?3.246
```

He now tries a value that is much too large.

```
60 ASN-ACS ARG ERROR
```

He gets an error message, and a return to editing/command level (where he will enter additional program statements for detecting the out-of-range condition).

```
>55 IF ABS(X) > 1 THEN 90
>90 PRINT X; "VALUE OUT OF RANGE"
>95 GOTO 70
>RUN
```

After inserting steps 55, 90, and 95, he tests again.

```
16:27    NOV 09 RUNIDAA...
COMPUTE ARCSINE OF X, IN DEGREES
```

Figure 8-1. Program Building, Editing, and Execution Using Basic (cont.)

```
?1.5
 1.50000  VALUE OUT OF RANGE
?  >

        He gets the desired result on the exception condition,
        and terminates execution.

!OFF

        (accounting summary)
```

Figure 8-1. Program Building, Editing, and Execution Using Basic (cont.)

## ENTRY OF JOBS TO THE BATCH JOB STREAM

In those instances where the on-line user does not wish to sit at the terminal and attend the execution of a long process, he or she may conveniently employ the terminal batch entry facility of CP-V to enter the job into the batch job stream for execution in the batch processing mode. The user may then disconnect from the system or start another time-sharing task.

This service allows time-sharing users to create and edit a control command file which will direct the execution of their jobs. At any time after submitting a job control file, the user may request the status of the job. CP-V responds by telling users the number of jobs ahead of theirs in the queue, that the job is running or that the job is completed. The user may also cancel the job from the on-line terminal.

Even if the batch mode is not operating concurrently with the time-sharing mode, jobs may be entered into the batch job stream for subsequent execution as soon as the batch mode is activated by the operations staff.

## COMMUNICATION WITH THE COMPUTER OPERATOR

Communication of control instructions to the CP-V operating system is accomplished through a Terminal Executive Language (TEL) by time-sharing users. Since the on-line user is in direct control of the computing task, the need for the vast majority of special instructions to the computer operator is eliminated. However, the need for some communication between the on-line user and the central operator still exists - for example, to request the mounting of tapes and disks or to request information.

CP-V provides facilities for the on-line user to transmit messages to the central site operator. When the message appears on the operator's console, the transmitting terminal is identified with the incoming message. The central operator can then carry on a dialogue with the individual on-line user.

For users not currently logged-on, the central operator may input a "greeting message". This message is stored by CP-V and is presented to the user immediately after logging on the system. Further information about the file management system is maintained automatically by CP-V. This information is placed in a "mailbox" and provides the user with current information on file disposition.

## AUTOMATIC PROCESSOR ASSOCIATION

The time-sharing mode allows the user to work at a terminal, interacting "directly" with a CP-V processor or with a user-written program. The word "directly" has been put in quotes because there are monitor routines which do not make themselves apparent to the user but which facilitate the interaction taking place.

In general, a time-sharing user may interact with a variety of processors during an on-line session. There is a feature of CP-V, however, which enables the system manager to restrict a user in such a manner that interaction may take place with only one selected processor. As soon as such a user logs onto the system, the user begins interacting with that one particular processor. The feature is particularly valuable when a user who is unfamiliar with CP-V is being introduced to the system or when a particular user requires only limited services. The basic purpose of this feature is to reduce the number of interactions at a terminal when only the one processor is going to be required.

## AUTOMATIC SAVE FOR LINE DISCONNECT

This feature of CP-V preserves a user's program when a line disconnect occurs before the user has logged off, and provides a method of reconnection of the preserved program when the user calls back. Files remain open and properly positioned so that the program may be continued as if it had never been interrupted.

When a line disconnect occurs, the suspended program image is retained for a fixed length of time. This retention period is established as a system parameter and may be modified by the operator at any time. (The operator may also abort a user when the user is in the suspended state.)

Suspended program images are retained and named by the user's log-on account/name identifier. Only one suspended image is retained for any given account/name. Thus, if two users are logged on under identical account/names and both hang up, only the image of the first to hang up will be retained. The second to hang up will simply be logged off. Further, the first to call back is given the option of reconnecting to saved image. Difficulty in this area can be avoided entirely by assigning unique account/names to each user.

When the disconnected user logs back onto the system, the system recognizes that a program image exists for his account/name combination and issues the following message:

```
    PROGRAM HELD. RECONNECT?
```

The user then responds with either Y or N. If Y, the user is reconnected to the suspended image and continues from the point of the disconnect. (However, I/O going to and from the terminal may have been lost.) If the response is N, the program image continues to be retained. (The retention time is not changed.)

## TERMINAL COUPLING

This facility provides for coupling (linking) of indirect printing mode terminals (e.g., Teletypes but not IBM 2741s) in such a way that the input and output of one terminal is displayed on both. All typing at both terminal keyboards appears on the paper of both. If the two terminal users are typing concurrently, then mixed (but identical) lines of characters appear on the two terminals. However, a running program of a particular terminal "sees" only the input of that terminal.

Conversations may be carried on between linked terminals by terminating lines with a cancel key-in so as not to affect a reading program. Terminal page heading output is not coupled. The link is broken if either line is disconnected.

This facility includes mechanisms for accepting, rejecting, creating, and terminating couplings. Both terminal commands and program procedures are provided so that the user may control coupling either using TEL commands typed at the terminal or through system procedures included within the program.

## PAPER TAPE INPUT

Paper tape may be used for input at Teletype terminals. There are three ways that a paper tape may be created:

1. The paper tape may be punched while the user has the Teletype in the local mode (i.e., not connected to the system).

2. The same characters that are keyed in during on-line input may be punched onto paper tape.

3. An existing file may be copied from system storage to the terminal with the paper tape punch on. Thus, the information being typed at the terminal will also be punched onto the paper tape.

# CHAPTER 9. MULTIPROGRAMMED BATCH PROCESSING

## OVERVIEW

CP-V offers a comprehensive multiprogramming batch facility for those jobs which do not need or do not benefit by on-line processing (time-sharing or transaction processing). The batch facility includes an easy to use, yet powerful, JCL structure. To increase throughput, up to 16 batch jobs may be run concurrently and a Multi-Batch Scheduler is provided that allows the installation manager to "tune" the system to meet the requirements of the installation. Also, a spooling facility (referred to as symbionts and cooperatives in CP-V) is provided to help eliminate bottlenecks associated with slow-speed peripherals.

## MONITOR CONTROL COMMANDS

### INTRODUCTION

When a job is submitted to the batch job stream, the user directs the operating system by means of a job control language (JCL) consisting of monitor control commands. These commands control the construction and execution of programs and provide communication between a program and its environment. The environment includes the monitor and processors (such as Meta-Symbol, COBOL, and ANS FORTRAN), the operator, and the peripheral equipment.

Most of the CP-V monitor control commands are briefly described in Table 9-1. There are additional monitor control commands which apply only to remote processing. They are discussed in Chapter 10.

Table 9-1. Monitor Control Commands

| Command | Function |
|---|---|
| AND | Causes a specified test to be made at a specified location. Only if the condition is true and the specified test identifier is set does it remain set; otherwise, it is reset (see SNAPC control command). |

Table 9-1. Monitor Control Commands (cont.)

| Command | Function |
|---|---|
| ASSIGN | Relates an operational label or a pseudo file name to a device. A pseudo file name may be assigned to an operational label. |
| BCD | Serves as a terminator for a binary input source. |
| BIN | Informs the monitor that the information to follow is binary. |
| COUNT | Specifies the range and the steps within the range where the test identifier is set (see SNAPC control command). |
| DATA | Informs the monitor that the information to follow is data. |
| EOD | Causes an end-of-data abnormal return to the monitor, indicating the end of a series of data records. |
| FIN | Specifies the end of a stack of jobs. |
| IF | Causes a specified test to be made at a specified location. The specified test identifier is set only if the condition is true; otherwise, the identifier is reset or remains reset (see SNAPC control command). |
| INCL | Directs the overlay loader to allocate public library routines in a segment. |
| JOB | Signals the completion of a previous job and the beginning of a new one. All jobs must have a JOB control command. |
| LDEV | Attaches an information stream to a physical device (identified by a logical device stream name) and defines attributes of the physical device. |
| LIMIT | Estimates the system job parameters (i.e., number of pages of output, number of cards to be output, time job is to run, etc.) for the job. |

Table 9-1. Monitor Control Commands (cont.)

| Command | Function |
|---|---|
| LOAD | Directs the Load loader to form a relocatable load module and enters it in the user's element file if a load module name is specified. |
| LYNX | Calls the Load loader to form a relocatable load module using a syntax compatible with Link. |
| MESSAGE | Causes the specified message to be typed to the operator at the time that it is encountered by the system. |
| MODIFY | Allows the user to insert a modification into a user program before execution. |
| NCTL | Allows noncontrol input files to be entered from the card reader. |
| OLAY | Equivalent to LOAD control command. |
| OR | Causes a specified test to be made at a specified location (if a specified test identifier is reset). If the condition is true, the specified test identifier is set; otherwise, it remains unchanged (see SNAPC control command). |
| OVERLAY | Equivalent to LOAD control command. |
| PFIL | Position n files on unlabeled magnetic tape. |
| PMD | Causes the monitor to dump the selected area of memory, in hexadecimal form, if an error occurs during execution. |
| PMDE | Causes the monitor to dump (in addition to the information obtainable by PMD) the PSD, registers, etc. |
| PMDI | Causes the monitor to dump the selected area of memory, in hexadecimal form, regardless of whether errors have been detected. |

Table 9-1. Monitor Control Commands (cont.)

| Command | Function |
|---|---|
| POOL | Tells the monitor the number of core pages to be allocated for buffers and tables associated with I/O operations. |
| processor name | Tells the monitor which processor is to operate and what options the processor is to execute. |
| PTREE | Tells the monitor that a tree control command is to be read from the user's file. |
| REW | Rewinds the specified tape. |
| RUN | Tells the monitor to transfer control to the user's program. |
| SNAP | Causes a snapshot of the specified memory and registers at the location specified to be performed. |
| SNAPC | Causes a snapshot of the specified memory and registers at the location specified to be performed only when the specified test identifier is set. |
| STEP | Provides conditional execution of job steps. |
| SWITCH | Produces the initial settings of the pseudo sense switches. |
| TITLE | Causes the specified title to be output at the beginning of each logical page of output on the listing output device. |
| TREE | Specifies the symbolic representation of the overlay structure. |
| WEOF | Writes a physical end-of-file on magnetic tape. |
| XEQ | Initiates processing of monitor control commands from a command file. |

## MONITOR CONTROL COMMAND EXAMPLES

To give the reader a greater understanding of the purpose and usage of monitor control commands, three examples of batch jobs are presented below.

In the first example of a batch job (Figure 9-1). a program written in Meta-Symbol assembly language is assembled and the object code is saved in a file called OBJECTCODE.

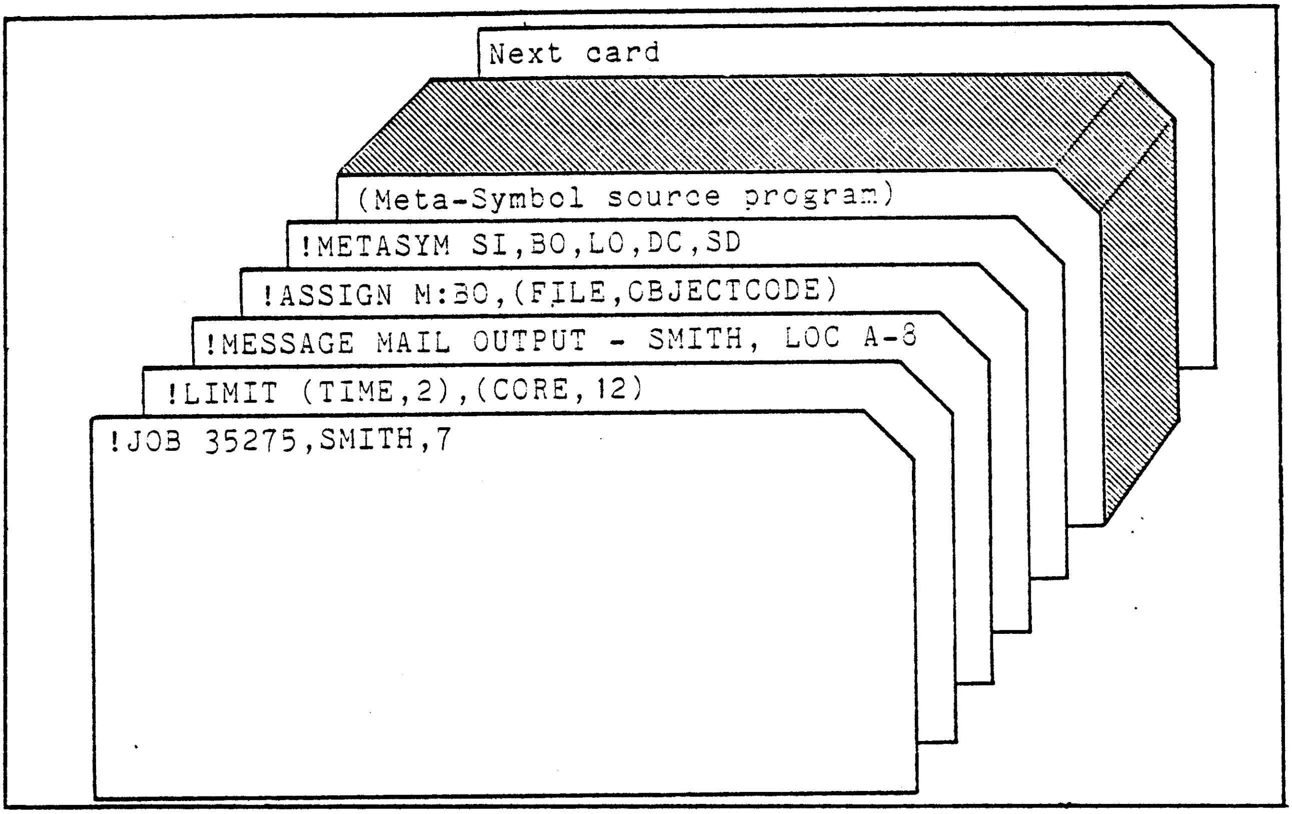

Next card
(Meta-Symbol source program)
!METASYM SI,BO,LO,DC,SD
!ASSIGN M:BO,(FILE,OBJECTCODE)
!MESSAGE MAIL OUTPUT - SMITH, LOC A-8
!LIMIT (TIME,2),(CORE,12)
!JOB 35275,SMITH,7

Figure 9-1. Assembly of a Meta-Symbol Source Deck

All cards beginning with an exclamation point are monitor control commands. In the example,

- o The JOB command identifies the user and specifies the priority at which the user is to run (7).
- o The LIMIT command specifies that the job is expected to require no more than two minutes of CPU time and 12 pages of core. (If either limit is exceeded, the job will be aborted.)
- o The MESSAGE command requests the operator to mail the output from the job to Smith at location A-8.

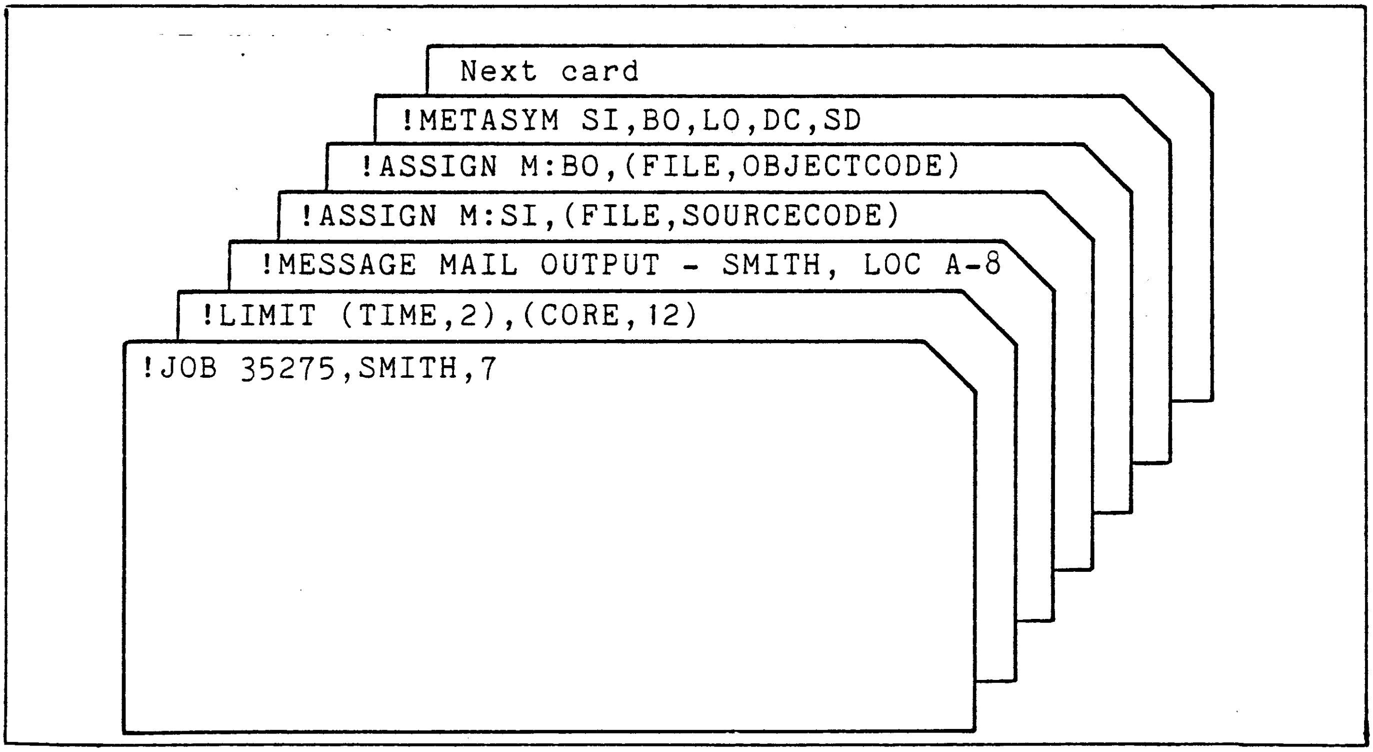
- o The ASSIGN command specifies that the object code is to be saved on a file called OBJECTCODE.

- o The METASYM command specifies that the Meta-Symbol assembler is to operate next and also specifies options that affect the operation of the assembler. For example, the SD option causes the assembler to produce symbolic debugging object code for use with the Delta debugging processor.

- o The Meta-Symbol source program is the program that is to be assembled.

- o The "Next card" could be any appropriate control command such as FIN or JOB.

In Figure 9-1, the Meta-Symbol source program appeared in the deck of cards submitted to the batch job stream. If the source program already resides in a file that is stored on some form of secondary storage, the Meta-Symbol assembler can be directed to access that file to obtain the source code. The example in Figure 9-2 is identical to that in Figure 9-1 except that the source program resides in a file called SOURCECODE (which, implicitly, is stored on system disk storage). Note that the entire job is defined by six monitor control commands.



```
Next card
!METASYM SI,BO,LO,DC,SD
!ASSIGN M:BO,(FILE,OBJECTCODE)
!ASSIGN M:SI,(FILE,SOURCECODE)
!MESSAGE MAIL OUTPUT - SMITH, LOC A-8
!LIMIT (TIME,2),(CORE,12)
!JOB 35275,SMITH,7
```

Figure 9-2. Assembly of a Meta-Symbol Source File

In the previous examples, the jobs have been prepared for entry into the batch job stream by punching them onto cards. If the user is at an on-line terminal, the job may be prepared through use of the Edit processor. Edit is used to build a file containing exactly the same information that would be punched onto cards. The file is then submitted to the batch job stream via a processor called Batch. Figure 9-3 shows the steps taken to prepare a job for the batch stream via this method. The reader should not be concerned with the details of the example, since the example is just intended to show the general process involved. When the BATCH JOBFILE command has been processed, the batch job stream will contain a job which is identical to that in Figure 9-2.

```
!BUILD JOBFILE
   1.000 !JOB 35275,SMITH,7
   2.000 !LIMIT (TIME,2),(CORE,12)
   3.000 !MESSAGE MAIL OUTPUT - SMITH, LOC A-8
   4.000 !ASSIGN M:SI,(FILE,SOURCECODE)
   5.000 !ASSIGN M:BO,(FILE,OBJECTCODE)
   6.000 !METASYM SI,BO,LO,DC,SD
   7.000

!BATCH JOBFILE
```

Figure 9-3. Submitting a Job to the Batch Stream from an On-line Terminal

# CHAPTER 10. REMOTE PROCESSING

## INTRODUCTION

The purpose of the remote processing system is to provide for flexible communication between CP-V and a variety of remote sites. Terminals at remote sites should not be confused with the on-line terminals which provide the direct user-computer interaction of the time-sharing mode. Remote terminals can range from a simple card reader and line printer combination to another large-scale computer system with an assortment of peripheral devices. Important features of the remote processing system include

- Support of a wide variety of peripherals at the remote site. Through monitor and user interfaces, virtually any type of device (e.g., tape, disk plotter) may be accessed with remote processing.

- Computer-to-computer communication. A remote site may be another large-scale computer, and files of data may be transferred between user programs at the central and remote computers.

- Slave/master status. A CP-V system may act as the central site to some remote terminals and as a remote terminal to other computers, simultaneously. This feature encourages the construction of communications networks.

- Complete user interface. Any user (batch, on-line, ghost) of a CP-V system can communicate with any number of devices at one or several remote sites. When data is being sent by a user program to a remote site, the remote site need not be connected since CP-V automatically buffers on RAD or disk for deferred transmission.

- Processing of jobs from remote sites at the central site. Jobs are sent from the remote site to the central site, are processed there, and may direct output to the originating remote site, a central site device, or another remote site as specified by the remote site user.

- Dynamic definition of remote stations. Remote stations can be added, deleted, or modified during system operation.

Basically, CP-V remote processing is a machine-to-machine communication mechanism that allows output files for pseudo devices called streams to be combined into transmission blocks and transmitted over communications lines. Blocks received over these lines are deblocked into symbiont input files for logical device streams (or real devices) at the central or remote site.

A block is the physical unit of data, including control information, transmitted between CP-V and a remote site. The block may consist of one or more logical records. There are many factors that influence the size of a block, including the terminal type. Generally, terminals that are capable of handling multiple records per block make more efficient use of communication circuits. CP-V provides the interface to both types of terminals. When appropriate, CP-V collects multiple records and transmits them in one block to a terminal. When receiving data from a remote batch terminal, CP-V deblocks the records. Thus a user program in CP-V sends or receives data in a common format regardless of the terminal being utilized.

## REMOTE PROCESSING TERMINALS

Two basic types of remote terminals are supported by CP-V: Remote Batch Terminals (RBTs) and Intelligent Remote Terminals (IRBTs).

An RBT is a card reader, and card punch, and line printer combination which is used primarily to allow batch processing I/O functions to be performed at remote sites. That is, a job is input to the system from the remote site card reader, the job is processed at the central site, and the output is sent to the remote site line printer or card punch. The output may optionally be directed to the central site or to another remote site. The Xerox 7670 RBT or any computer that exactly emulates it (Univac DCT 2000 compatible) is supported by CP-V.

The IBM 2780 RBT or 3780 RBT, or any computer that exactly emulates them, is supported by CP-V given the following factors:

- o EBCDIC transmission code.
- o Nontransparent line protocol.
- o Single record blocks or multirecord blocks of 400 bytes for 2780 RBTs and 512 bytes for 3780 RBTs.
- o Support for multidrop lines is not provided.

An IRBT can be either a mini-computer system for which the primary function is to control the operation of peripheral equipment (e.g., COPE 1200) or another large-scale computer system (e.g., another CP-V system). Any computer system that supports the IBM HASP Multileaving protocol may act as an IRBT to CP-V. (This and all other reference to "HASP" and "Multileaving" in this document refer to the HASP Multileaving protocol as described in HASP Version 2.3 program documentation and not to the IBM HASP operating system, except where IBM HASP is specifically noted.) Multileaving allows a single data block to contain records associated with different peripherals at the IRBT. In conjunction with a feedback mechanism that temporarily suspends transmission for a single peripheral, multileaving permits peripherals of different speeds to operate at their individual rates.

The majority of records transmitted to and from any computing system contain many adjacent characters that are identical. CP-V and supported IRBT systems increase throughput by contracting (compressing) strings of characters before transmission and expanding (decompressing) these character strings after receipt. Thus, while the compression/decompression process improves overall system efficiency, the user of transmitted data does not have to be aware of its source or format.

## HARDWARE CONNECTION OF REMOTE TERMINALS

A remote terminal is connected to the central site over a communication line that is either a hardwired line or a switched line.

If the connection is over a hardwired line, the remote site must be physically near the central site.

If the connection is by switched line, two data sets (modems) are required. The data sets provide interfaces between the line and the remote terminal and between the line and a data set controller (described below) at the central site. The data sets convert device signals to telephone tones and telephone tones to device signals.

All remote terminals require a data set controller (DSC) for interface with the central site. A virtually unlimited number of RBTs and IRBTs may be connected to the computer via a particular DSC, but only one may be connected at a time. Therefore the maximum number of remote sites that may be connected concurrently is determined, not by the number of remote terminals, but by the number of DSCs at the central site. CP-V supports up to 30 DSCs of the following types:

Xerox 7601, which may be used only with a Xerox 7670 RBT.

Xerox 7605, which may be used with any RBT or IRBT.

Options that may be added to either of the above DSCs are

Xerox 7602, which provides full-duplex operations.

Xerox 7604, which provides for local connection.

Figure 10-1 depicts the relationship between the central site, remote sites, data sets, and data set controllers.

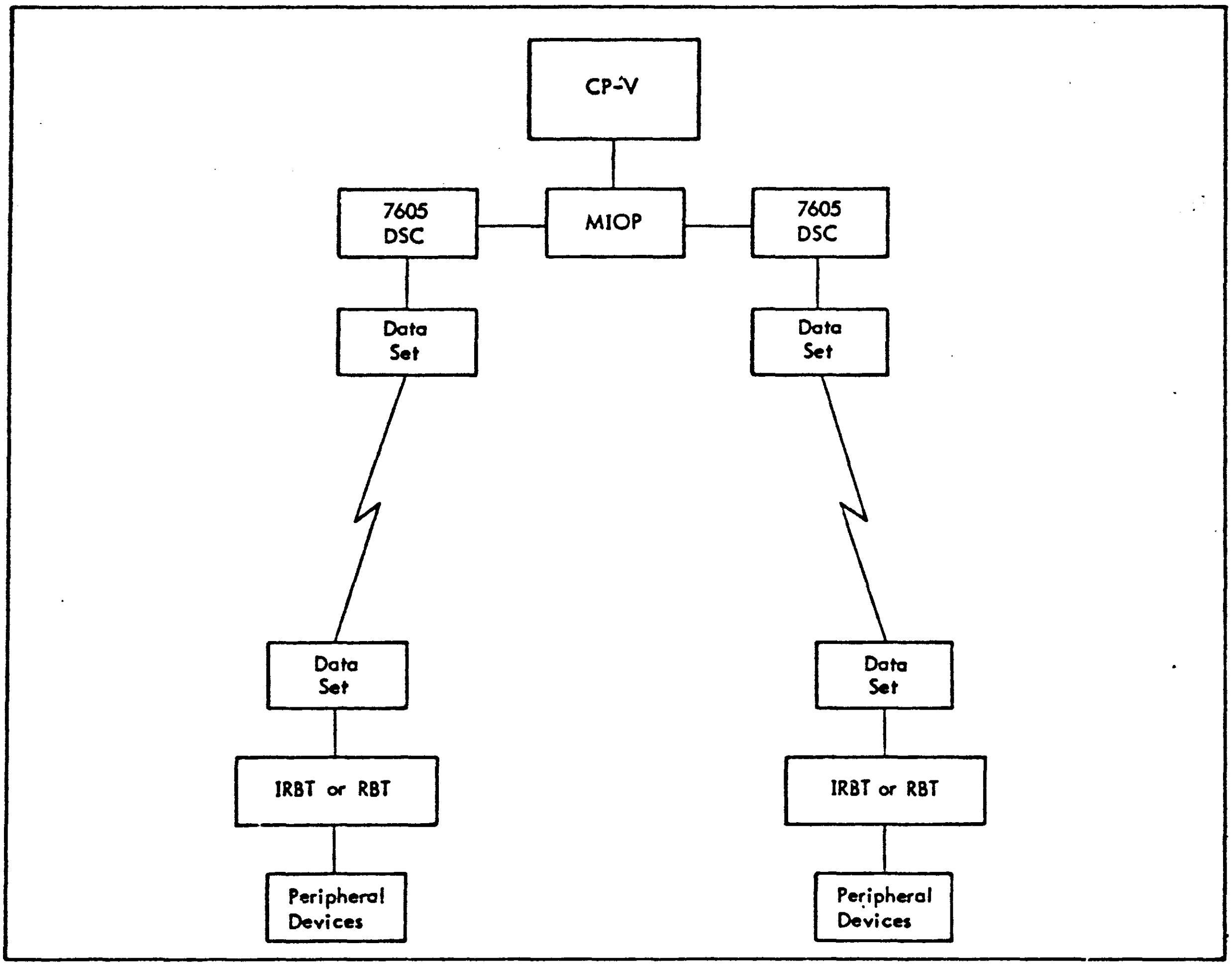


Figure 10-1. CP-V Remote Processing Hardware

## REMOTE PROCESSING MODES

The remote processing system is designed so that the CP-V system may act as the central site to a set of remote terminals while simultaneously acting as a remote terminal to one or more other systems. A system that is acting as the central site is referred to as the "master" system and a system that is acting as a remote terminal is referred to a "slave" system.

To the CP-V system, the role of master and slave manifests itself only at log-on time. After the data-set to data-set communication path is established, the slave logs onto the master. (The master cannot log onto the slave.) Once the log-on is complete, the communication path between the master and slave is symmetrical - streams of data flowing in both directions over the communication path.

Four fundamental modes of remote processing are

1. A CP-V master system connected to one or more slave Xerox 7670 and/or IBM 2780 RBTs.

2. A CP-V master system connected to one or more slave mini-computer IRBTs.

3. A CP-V system communicating with another CP-V system.

4. A CP-V system acting as a slave IRBT connected to another computer system acting as the master computer.

These four modes may be combined to provide a large variety of communications networks. An example of such a network is given in Figure 10-2. (The arrows point to the RBTs and slave IRBTs.)

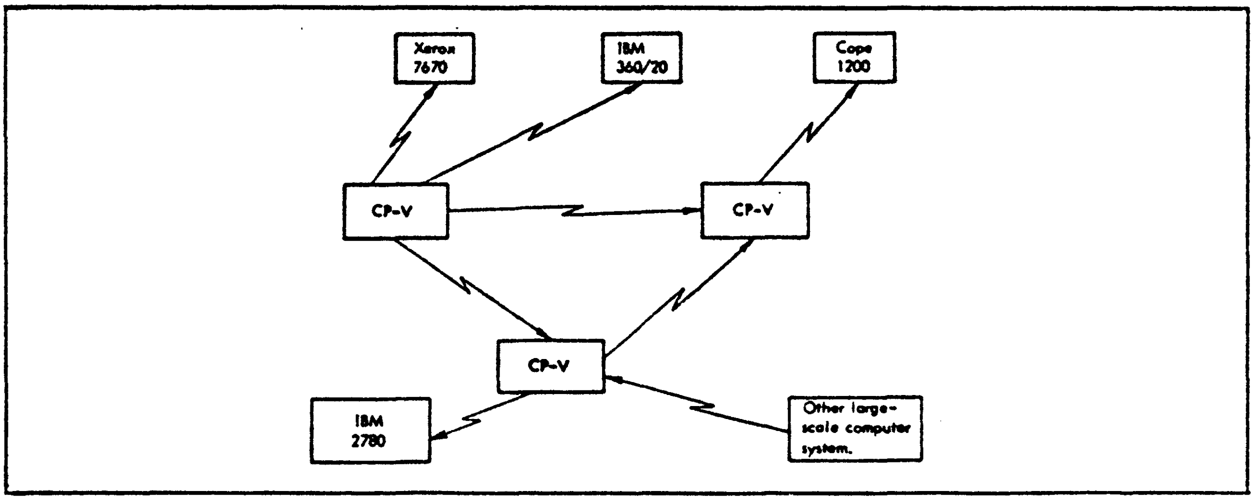


Figure 10-2. CP-V Remote Processing Network

## WORKSTATIONS

A workstation is a named entity used to represent a set of hardware characteristics at remote sites. (In sophisticated applications, a workstation definition may include the characteristics of pseudo-hardware.) The workstation may represent all of the hardware at a given remote site, the hardware available to a particular group of users at a given remote site, or the hardware available to a particular group of users at any site which has the requisite configuration. The definition of a workstation specifies such items as:

- Name of the workstation.
- Type of terminal to be used (RBT or IRBT).
- Maximum priority for jobs submitted from the workstation.
- Whether the workstation will be a slave computer or the master computer (if the type is IRBT).
- Remote peripheral devices to be associated as part of the workstation (if it is an IRBT).
- Attributes of devices defined for the workstation.

Each workstation is given a workstation name (WSN) of one to eight alphanumeric characters. Local devices at a CP-V system have the workstation name LOCAL.

A workstation is not limited to use at one physical location. A user may log on as a given workstation at any remote site that has the appropriate hardware characteristics. Each remote site may have several workstations defined for its use, but only one workstation may be active on a given line at a given time. If the remote site has more than one line, then more than one workstation may be logged on concurrently from that site. The workstations must have different WSNs, however, because a given WSN cannot be in concurrent use.

A workstation definition may be created dynamically any time during system operation by the system manager. The attributes of workstations may also be changed and workstations may be deleted during system operation.

## MONITOR CONTROL COMMANDS

There are several monitor control commands that pertain only to remote processing. These commands are summarized in Table 10-1. These control commands facilitate the use of remote processing terminals. The function of transferring files of data between machines is performed via the ISCL processor and two other monitor control commands, JOB and LDEV.

Table 10-1. Remote Processing Monitor Control Commands

| Command | Function |
|---|---|
| RBID | Logs a workstation onto the system. |
| RBDISC | Logs a workstation off the system. |
| RBXXX | Logs a workstation off the system and disconnects it immediately. |
| RBMSG | Transmits a message to the local operator. |
| RBDEV | Displays the status of all devices at the workstation and, where applicable, the name of the form mounted on each device. |
| RBINFO | Displays various CP-V statistics. |
| RBPRIO | Changes the priority of the workstation's files in the symbiont system to a specified priority. |
| RBHOLD | Prevents current output files and output files from other sources from being output, but does not affect input files (except that results from the execution of such files is held). |
| RBRETRIEVE | Releases files that were held with RBHOLD. |
| RBDELETE | Deletes input, output, or executing files from the symbiont system. |
| RBSTATUS | Requests the status of files belonging to the workstation. |

Table 10-1. Remote Processing Monitor Control Commands (cont.)

| Command | Function |
|---|---|
| RBSWITCH | Changes the workstation assignment of output files. |
| RBSUSPEND | Suspends output on specified device(s). |
| RBCONTINUE | Specifies that suspended output is to be continued from where it stopped. |
| RBREPRINT | Restarts output at the symbiont retry point. |

## ISCL PROCESSOR

When the remote site is another CP-V system, the Inter-System Command Language processor (ISCL) facilitates manipulation of CP-V managed files for both local and remote users. The ISCL processor provides the on-line or batch user with the ability to copy, create, list or delete files in another CP-V system via an IRBT connection.

The processor interprets the user ISCL command and packages the command and possibly a file for transmission to the remote CP-V system. A simple file transmission is carried out in three stages. First, the file is copied to the symbiont disk file, the staging area for all remote transmissions. Second, after establishing a communication channel to the remote site, all files of information destined for that site are blocked into transmission packages. At the receiving site, the packages are deblocked and each information file is recreated on disk. Finally, a ghost job is initiated at the remote site (as a consequence of the file's type) which copies the symbiont file to the proper, cataloged destination file as per the original instructions. When the file is built in the remote CP-V system, a message indicating addition of a file to the remote account is placed in the MAILBOX file of the remote account.

To use the ISCL processor, the user must be authorized at the remote site as well as the local site. User requests may

o Copy a file from a file at a remote CP-V system.

o Send a file to a file at a remote CP-V system.

- Delete a file at a remote CP-V system.
- List file(s) which are at a remote CP-V system.

## LDEV AND JOB CONTROL COMMANDS

Two commands, LDEV and JOB, are the other primary methods that local and remote users use to take advantage of remote processing capabilities.

These are the two CP-V commands that have options that allow a user at one site to direct a file to another site. The LDEV command allows users to direct a stream of information anywhere within the reaches of remote processing - to a symbiont device at the central site or to any device at a specific remote workstation. It also allows users to direct files to other sites to be run as jobs in the batch processing mode.

The JOB command, with a more limited application to remote processing, provides a means of directing the print and punch output of the job to a specified workstation or to the central site.

It is important to note that the remote user may submit jobs to the central site that make full use of the LDEV and JOB command remote processing features provided that the central site is a CP-V system. Because of the on-line/batch compatibility of CP-V, the two commands may be used by both on-line and batch users. In cases where the master computer is a system other than CP-V, the remote user must be familiar with the remote processing commands and capabilities of the particular master system.

# CHAPTER 11. REAL-TIME PROCESSING

## REAL-TIME PROGRAMS

Real-time processing involves reacting to external events (including clock pulses) within microseconds. Selected external events are allowed to interrupt the real-time user's program so that they can be processed at the time they occur. After an interrupt has been processed, control may then return to the interrupted program or may be directed elsewhere.

In CP-V real-time processing, there are three distinct types of interrupts:

1. Real, hardware interrupts from devices wired to the CP-V computer.

2. Multiple clock interval interrupts derived through software from a single hardware clock interrupt.

3. User written pseudo-interrupts that are triggered by software rather than by hardware. This type of interrupt is quite useful for interprogram communication and synchronization.

There are two major categories of CP-V real-time programs: unmapped and mapped. Basically the unmapped category provides fast, limited service facilities and the mapped category provides slower responding, full service facilities.

Mapped real-time programs begin execution as normal on-line or batch programs or as a special type of program called ghost jobs. They are known to the CP-V execution scheduler and have their interrupt connected such that events causing an interrupt are reported to the CP-V execution scheduler. This is called central connection. Mapped programs are normally subject to swapping, but may lock themselves in memory if required. All monitor services are available to mapped real-time programs. Monitor services provided specifically for mapped real-time programs include the following functions:

- o Interrupt connection/disconnection, statusing, and control.

- o Program execution and priority control.

- Locking the program in memory (i.e., prevent it from being swapped).

- Clock wake-up services.

- Device preemption and return.

- Direct I/O Services.

- Changing the Master/Slave bit in the Program Status Word (PSW). (This bit controls whether the program can execute privileged hardware instructions or not.)

- Changing the memory map by altering the correspondence of virtual addresses to physical memory addresses.

Mapped real-time programs may be connected to either hardware interrupts or software simulated peudo-interrupts. Response time for these programs is approximately 1 millisecond. In a multi-processing system, mapped real-time programs can be executed on any available execution resource unless they execute in the master mode. Master mode programs can only be executed by the primary processor. (In a multi-processing system, one execution resource is designated the primary processor. The primary processor handles all interrupts, monitor services, and performs all monitor functions (e.g., execution scheduling, memory management). It can also execute user program code. (The remaining execution resources are called secondary processors and can only execute slave mode program code.)

The second category, unmapped real-time programs, is considerably different. Such programs receive control directly from the interrupt without any intervention from CP-V; in fact, CP-V is unaware of the occurrence of the interrupt. This is called direct connection to an interrupt. These programs are loaded into special pages of real memory reserved by the operator. They do not use the memory map and essentially become extensions to CP-V. Maximum response to events, usually (98% of the time) less than 500 microseconds, is achieved using unmapped real-time programs. Since these programs receive the interrupt directly, they cannot use monitor services. They can, however, branch directly into CP-V to make use of some monitor routines. This includes the facility to:

- Obtain/release real pages of memory.

- Obtain/release granules of secondary storage.

- Use the CP-V basic I/O routines.

- Send character(s) to a time-sharing user terminal.
- Initiate a ghost job.
- Report a user event to the execution scheduler.

Unmapped real-time programs can only be connected to hardware interrupts. In a multi-processing system, unmapped real-time programs can only be executed by the primary processor.

Most real-time programs require an orderly shut-down in the event of a system or power failure. To accommodate this need, CP-V passes control to installation-provided routines during the processing of failure conditions. Also, as part of the recovery process from any failure, CP-V automatically passes control to installation supplied routines so they can re-initialize the real-time activities as needed.

## REAL-TIME PROCEDURES

Table 11-1 briefly summarizes the system procedures which are designed specifically for real-time processing. The following terms appear in discussions of real-time services:

### Disarmed

When an interrupt is in the disarmed state, no signal to that interrupt is admitted; that is, no record is retained of the existence of the signal, nor is any program interrupt caused by it at any time.

### Armed

When an interrupt is in the armed state, it can accept and remember an interrupt signal. The receipt of such a signal advances the interrupt to the waiting state.

### Waiting

When an interrupt in the armed state receives an interrupt signal, it advances to the waiting state, and remains in the waiting state until it is allowed to advance to the active state.

### Enabled

When an interrupt is in the enabled state, it is allowed to move to the active state when the interrupt signal is received provided that it is also in the armed state. If the interrupt is already in the waiting state, it moves to the active state when it becomes enabled, provided that no higher priority interrupt is currently active.

### Disabled

An interrupt can undergo all state changes except that of moving from the waiting to the active state when it is in the disabled state.

### Active

When an interrupt meets all of the conditions necessary to permit it to move from the waiting state to the active state, it is permitted to do so by being acknowledged by the computer, which then executes the contents of the assigned interrupt location as the next instruction.

### Cleared

When an interrupt is changed from the active state to the cleared state, the interrupt states are reset so that the interrupt can be recognized again and the priority is reset to that of the job that was running when the interrupt occurred.

### Interrupt Control Blocks (ICBs)

ICBs are areas of memory set aside for use by the monitor interrupt processing routines. ICBs are established by SYSGEN.

### Interrupt Label

The two-character name of an interrupt. Interrupt labels are defined at SYSGEN.

Table 11-1. Real-Time Procedures

| Procedure | Function |
|---|---|
| M:GJOBCON | Associates an interrupt with a load module such that if the interrupt occurs, the designated load module will be put into execution as a ghost job. |
| M:CONNECT | Establishes a connection to an interrupt such that the user program will be entered at a specified address when the interrupt occurs. This procedure is only available to mapped user programs. Interrupts connected in this way report events to the CP-V execution scheduler and therefore permit the entered program to use all monitor services. |
| M:DISCONNECT | Releases the specified interrupt if it is associated with the user. |
| M:INTCON | Permits a program to control the states of interrupts. Interrupts may be armed, disarmed, enabled, disabled, or triggered. |
| M:INHIBIT | Permits a program to prevent itself from being interrupted by any higher priority real-time task. |
| M:INTRTN | Allows a mapped, scheduled program entered as the results of a centrally connected interrupt or elapsed clock interval to return to the point of interruption. The actual return is to the environment that existed for this program or user when the interrupt occurred even if this user was not in control when the interrupt occurred. |
| M:QFI | Permits the user to suspend execution while awaiting interrupts or elapsed clock intervals assigned a priority higher than the current execution priority. If there are no interrupts connected for this user that satisfy this condition, the user is aborted. |
| M:INTSTAT | Permits any user to query the status of any real or pseudo-interrupt location. |

Table 11-1. Real-Time Procedures (cont.)

| Procedure | Function |
|---|---|
| M:HOLD | Prevents a program from being swapped. |
| M:CLOCK | Permits a user with a sufficient privilege to request entry at a specified address when a specified time interval has elapsed. |
| M:STOPIO | Obtains direct control over the I/O associated with a particular device and ensures that there will be no contention for a particular device during certain critical processing periods. This includes the ability to request I/O end action off of the I/O interrupt associated with the I/O operation. |
| M:STARTIO | Returns an I/O device preempted via M:STOPIO to the system. |
| M:IOEX | Provides a means of enabling the real-time user to exercise direct control over I/O operations without having to run in the master mode (see also the M:EXU service). The only requirements are that the device specified be preempted (either via the M:STOPIO service or the SYSCON processor), and that an end-action routine be provided (either via M:STOPIO or M:IOEX). The I/O functions that can be controlled via M:IOEX are:<br>SIO - Start input/output.<br>HIO - Halt input/output.<br>TIO - Test input/output.<br>TDV - Test device. |
| M:EXU | Provides another way for the real-time user to execute I/O instructions and other privileged instructions without having to run in the master mode (see also the M:IOEX Service). The only requirement is that the |

Table 11-1. Real-Time Procedures (cont.)

| Procedure | Function |
|---|---|
| M:EXU (cont.) | instruction op code to be executed be one of the following:<br>Op Code — Mneumonic<br>X'4C' — SIO<br>X'4D' — TIO<br>X'4E' — TDV<br>X'4F' — HIO<br>X'6C' — RD<br>X'6D' — WD<br>The SIO execution service is intended primarily for interfacing to devices not known to the operating system and which do not generate I/O interrupts. However, no validity checks are made and if the SIO will result in an I/O interrupt, it is assumed that the user will have provided an end-action receiver via the M:STOPIO service. |
| M:MASTER | Allows a user with sufficient privilege level to operate in the master mode (master-protected mode if running on a Sigma 9 or Xerox 560) with a write key of 1. |
| M:SLAVE | Allows any master (and master-protected) mode program to return to the slave mode. |
| M:MAP | Converts a specified virtual address to a physical address or a specified physical address to a virtual address. |
| M:GPP | Acquires a physical page of memory. |
| M:FPP | Releases a physical page of memory that was acquired by M:GPP. |
| M:GJOB | Activates (or awakens) a program as a ghost job. |

Table 11-1. Real-Time Procedures (cont.)

| Procedure | Function |
|---|---|
| M:GDG | Acquires a disk granule dynamically. |
| M:RDG | Dynamically releases a disk granule acquired via M:GDG. |
| M:RUE | Simulates that an event took place for the user (e.g., an error, wake-up, log-off). |
| M:CHKINT | Checks the status of an interrupt. |
| M:EXCP | Executes the user's own channel program. (This procedure results in a BAL linkage to the monitor's I/O supervisor module.) |
| M:NEWQ | Requests I/O to be performed without a user-built channel program. (This procedure results in a BAL linkage to the monitor's I/O supervisor module.) |
| M:QUE | Requests that I/O be performed through parameters supplied in a specified DCB. (This procedure results in a BAL linkage to the monitor's I/O supervisor modules.) |
| M:COC | Sends a character to the user's terminal. |

## DYNAMIC PHYSICAL PAGE ALLOCATION

Physical pages are made avalable for real-time processing in either of two ways:

- o Dedication of physical core pages at boot-time. These pages are known as the Resident Foreground (RESDF) pages. Parameters specified during system generation define the physical pages that are to be removed from the system and dedicated to real-time processing. These pages remain dedicated real-time pages until returned to the system via the Physical Page Stealer (PPS) Ghost.

- o Dynamic acquisition and release of physical core pages during normal operations. These pages are known as the

Dynamic Resident Foreground (DYNRESDF) pages. The operator can acquire or release DYNRESDF pages by communicating with the Physical Page Stealer (PPS) ghost job.

In both cases, foreground memory is allocated in 'memory segments'. A memory segment in this context is simply a set of contiguous physical pages. There is only one RESDF memory segment (i.e., that which may be allocated at boot-time). There may be several DYNRESDF memory segments, the maximum number of which is specified during system generation (SYSGEN). All real-time memory segments must be allocated in the area between 64K and the end of physical main memory.

The operator, by communicating with the Physical Page Stealer ghost job, has control over the allocation of both RESDF and DYNRESDF pages. The operator also has the ability to reset the SYSGEN defined RESDF size and maximum DYNRESDF size thus affecting the system's maximum user size. Increases to RESDF size or to maximum DYNRESDF size cause a decrease of the maximum user size; decreases to RESDF size or the DYNRESDF size cause the maximum user size to be increased. By setting the maximum number of real-time pages that may be allocated to a minimum, the operator is able to allow very large jobs to be scheduled. Decreases to the maximum real-time page values may be effected at any time. Increases that would cause the maximum user size to be set to less than 186 pages are limited to times when there are no users on the system other than system ghosts; i.e., the system must be quiescent except for certain system ghosts. Neither RESDF nor DYNRESDF maximum size may be increased to the point where the maximum user size is too small to allow the system ghosts to run.

# CHAPTER 12. TRANSACTION PROCESSING

## INTRODUCTION

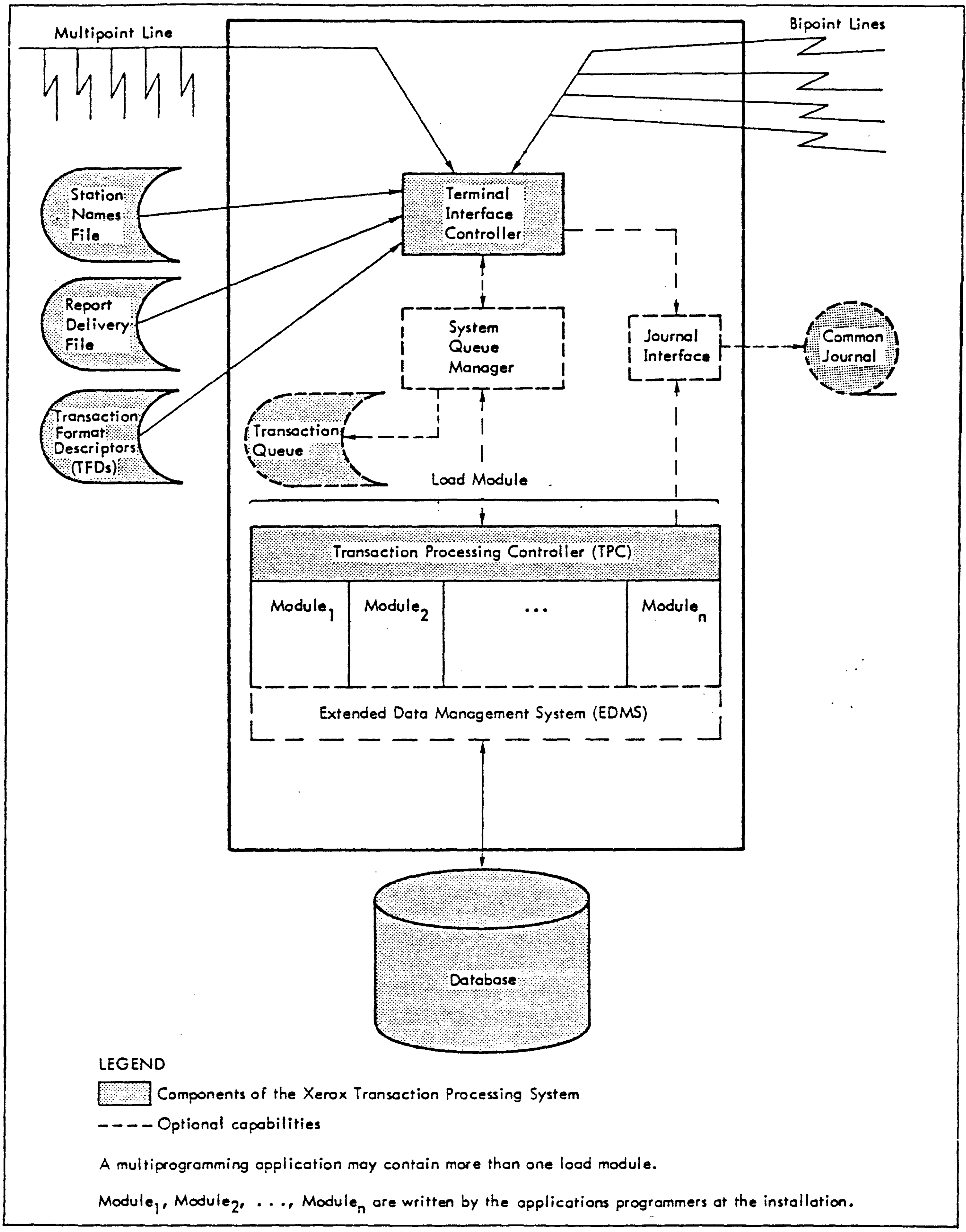
Transaction processing (TP) is a recent innovation of the computer industry. It is a unique combination of hardware and software designed to satisfy the data processing requirements of business and the professions. Transaction processing combines the hardware field of teleprocessing with the software fields of operating systems, data management systems, and special transaction processing components. Critical to the entire operation is teleprocessing which allows information to be transmitted between a computer center and user stations located in the surrounding areas.

In the software realm, transaction processing requires the standard operating system functions of job queuing, scheduling, and execution. It also requires the current sophisticated data management systems that have evolved from the simple storage and retrieval of information. This generalized software is further enhanced with transaction processing components that interface with user stations and with software designed at an installation. The result of this hardware and software combination allows business and the professions to query a central database or to dynamically modify its contents as transactions occur throughout the organization.

Components of CP-V transaction processing appear in Figure 12-1. The shaded portions denote transaction processing software components and files which are part of the CP-V system. The unshaded blocks are modules developed by applications programmers at the installation. The Extended Data Management System (EDMS) is part of the CP-V system but its use in transaction processing is optional.

Because the Transaction Processing Controller and the utility processors that create the Station Names, Report Delivery, and TFD files execute as independent CP-V jobs, an installation may replace any component with its own general-purpose or specialized program, or it may use its own programs in addition to those provided with CP-V.

All inputs to TP are transactions and all outputs are reports. While it is possible to enter a transaction that does not produce a report, it is not possible to obtain a report without a transaction to cause its creation. Hence, the entry of a single



LEGEND

▨ Components of the Xerox Transaction Processing System

---- Optional capabilities

A multiprogramming application may contain more than one load module.

$\text{Module}_1$, $\text{Module}_2$, . . ., $\text{Module}_n$ are written by the applications programmers at the installation.

Figure 12-1. CP-V Transaction Processing

transaction may produce no reports, a single report, or several reports, depending on the design requirements for the software module that processes the given transaction type.

## TERMINAL INTERFACE CONTROLLER

The Terminal Interface Controller is of particular interest to the system manager, whose decisions are relayed to this component, and to the central operator, who regulates it operationally using a group of commands. During its initialization phase, the Terminal Interface Controller reads the external control files - Station Names, Report Delivery, and TFD (described below) - and thereby establishes the current transaction processing environment. Once initialized, the Terminal Interface Controller acts as a complete system interface for all stations in the communications network. For example, it logs user stations on and off, verifies passwords, accepts transactions and delivers reports, and acts on commands entered by station users. Thus, the Terminal Interface Controller accepts transactions in a multiplexed fashion from the stations it controls and routes the resulting reports back to the stations.

### STATION NAMES AND REPORT DELIVERY FILES

The Terminal Interface Controller reads the Station Names File in order to identify stations that it may control during the course of transaction processing. This file associates logical names with physical stations, specifies any required passwords, and may restrict the types of transactions entered from certain stations. A logical name is one entered by a station user as he logs onto the system. A logical name may also identify a station to which a report is to be delivered, even though the transaction that produced the report did not originate at that station. The Station Names File is created by the Station Names Processor.

A report may be sent back to the station that originated the transaction, or a report may be routed to the location of another station in the communications network. Whenever a report is to be delivered "to location", the Terminal Interface Controller requires the delivery instructions for the report. Delivery instructions are specified by using the Report Delivery Processor to create the Report Delivery File.

## TFD FILE

Transaction Format Descriptors, called TFDs, describe the form of transactions and reports. Although differences exist, a TFD is similar to a PICTURE clause in COBOL and to a FORMAT statement in FORTRAN. At least one TFD must be prepared for each transaction type and for each report type in the system. For transactions, TFDs are used to detect errors typically made by a station user, such as entering data in the wrong field and omitting required information. For reports, TFDs control the report layout (titles, columns, rows), meaning that applications programmers are not required to format reports. The TFD Processor, which generates this file, permits TFDs to be inserted, modified, or deleted as required by the application.

## OPERATOR CONTROL

The central operator typically invokes the Terminal Interface Controller at the start of each transaction processing day and shuts it down at the end of the day. Commands provided for operational control allow him to enable or disable the entire communications network, to acquire stations into the network or to free them, and to shut down the Terminal Interface Controller itself.

## COMMON JOURNAL

Transactions and reports may be recorded in the common journal as historical records, for data collection purposes, or for recovery in the event of a system malfunction. Use of this journal is optional. When used, it provides a complete audit trail of the flow of transactions and reports through the system, with the date and time stamped on each record. If the installation elects to journalize transactions or reports, or both, then in the event of a malfunction, any affected transactions and reports may be reprocessed as required from the common journal.

A system may include several journals, but only one can be associated with the transaction queue.

## SYSTEM QUEUE MANAGER

The System Queue Manager is part of the CP-V monitor. It manages the transaction queue, which is a standard random access file, by accepting transactions from the Terminal Interface Controller and queuing them until the Transaction Processing Controller requests them. Conversely, the System Queue Manager accepts reports from the Transaction Processing Controller and queues them until they are requested by the Terminal Interface Controller.

The transaction queue is defined to, and owned by, the TP Ghost (discussed below). Execution of the System Queue Manager is started by the TP Ghost and thereafter all queuing functions are completely automatic. Moreover, the System Queue Manager is a generalized procedure and its queuing capabilities may be used for purposes other than transaction processing.

When reports and transactions have been completely processed, the System Queue Manager deletes them from the queue. This technique frees the queue space and reduces the probability of "losing" transactions or reports within the system.

In typical TP usage, all transactions and reports are automatically queued, but the System Queue Manager may be bypassed as shown in Figure 12-1. Queuing is optional because the system may be used in nontypical ways, some of which are discussed toward the end of this chapter.

## TRANSACTION PROCESSING GHOST

The TP Ghost (not shown in Figure 12-1) is a system component that must be explicitly started by the central operator. Once started, the operator may define system files, open and close files, and terminate transaction processing.

Specifically, the TP Ghost maintains a file named TPFILES that describes the transaction queue, the common journal associated with the queue, and any other journals in the system. The TP Ghost allocates and initializes these files. Using a special set of commands, the central station operator may set file definitions in TPFILES; delete definitions from TPFILES; open, close, and release files; switch journal volumes; and terminate transaction processing.

## TRANSACTION PROCESSING CONTROLLER

The Transaction Processing Controller (TPC) is of interest to applications programmers at the installation: the actual processing of transactions and the creation of reports occurs in user modules which they write in COBOL, Meta-Symbol, Assembly Program, or FORTRAN. In effect, the TPC consists of a main program and several subroutines. The TPC main program performs service and control functions for user modules by requesting transactions from the System Queue Manager, accepting and queuing reports created by user modules, and journalizing reports as designated.

TPC subroutines are called by the user modules. For instance, the INITATPC subroutine initializes a user module and specifies the types of transactions that module is designed to process; the GETATRAN subroutine obtains the text of the current transaction; and the OUTALINE subroutine supplies the TPC main program with the report contents.

As many as 100 user modules may be linked together with a TPC to form a load module. Whenever a central database is managed with the Extended Data Management System, the EDMS Database Manager is also included in the load module for data manipulation.

## OPERATOR CONTROL

Two-way communication exists between the TPC and the central operator. Using commands, the operator may regulate user modules on and off or shut down the load module itself, and the TPC is automatically notified. Conversely, because of some abnormal processing condition, the TPC may abort the load module or one of the user modules, in which case it informs the operator of the condition.

## TPC SIMULATOR

As each user module is written, the programmer may test it with the TPC Simulator before executing that module in a live transaction processing environment. In particular, use of the TPC Simulator can prevent untested modules from affecting the common journal and the central database. The TPC Simulator - a version of the TPC - is generated from the TPC source program.

# EXTENDED DATA MANAGEMENT SYSTEM

The Extended Data Management System (EDMS) is designed especially for organizations that require the same data for many purposes, by many different departments. An EDMS database consists of network-structured information required for different activities, such as payroll, personnel and inventory. EDMS consists primarily of a Data Definition Language, a File Definition Processor, the Database Manager, and utility processors. The EDMS User's Guide is publication number 90 30 37; the EDMS Reference Manual is publication number 90 30 12.

## DATA DEFINITION LANGUAGE

The database administrator employs the Data Definition Language to describe the desired database structure in terms of items, groups, and sets. Because sets relate data items, they are the most important feature of EDMS. Sets relate data such that it is physically stored only once, even though that data may be retrieved or modified later by several different departments. For instance, an employee's social security number, salary, and hire data usually appear in the employee's payroll record and also in his personnel record. Because of set relationships, this data appears once in the database, but the payroll department and the personnel department each have that data in their required form, without an artificial interface.

After defining the structure with the Data Definition Language, the database administrator uses the definition as input to the File Definition Processor. This processor creates the file definition in an internal format required to initialize the database. The actual database may then be created.

## DATABASE MANAGER

EDMS may also be used by applications programmers. Once the database has been created, programmers can manipulate the data using the Database Manager - the term given to a group of library routines. These routines may be called by user modules to open and close the database, to find a group, to get an item, to store a new occurrence, and to delete an occurrence. The Database Manager is never used to modify the structure originally defined by the database administrator.

## CHRONOLOGICAL SEQUENCE OF EVENTS

Figure 12-2 summarizes the foregoing description of the CP-V Transaction Processing System. It lists - in chronological order - the events that occur during transaction processing and shows the paths of transactions and reports as they flow through the system. Table 12-1, which follows the illustration, expands the annotation that appears adjacent to the event sequence numbers. The common journal records named in the figure are defined later in this chapter.

## SYSTEM INTEGRITY

The integrity of the central database, and therefore of the entire system, depends on the inclusion of specific records in the common journal. When journalization of certain records is specified, other records are automatically journalized as diagrammed in Figure 12-3. (All common journal records are listed and defined in Table 12-2.)

If an installation elects to journalize transactions or reports, or both, then transactions and reports being processed when a failure occurs may be reprocessed from the common journal. Should a failure occur, the database administrator must decide which of the recovery phases to execute and the central operator initiates the recovery process.

## RECOVERY

Recovery is an important aspect of transaction processing since the system must be capable of detecting failures and of recovering from them. Some failures that can occur are operator errors, accidental destruction of the database or of the transaction queue, and violation of rules established for system usage. Errors may also occur in user modules written at the installation.

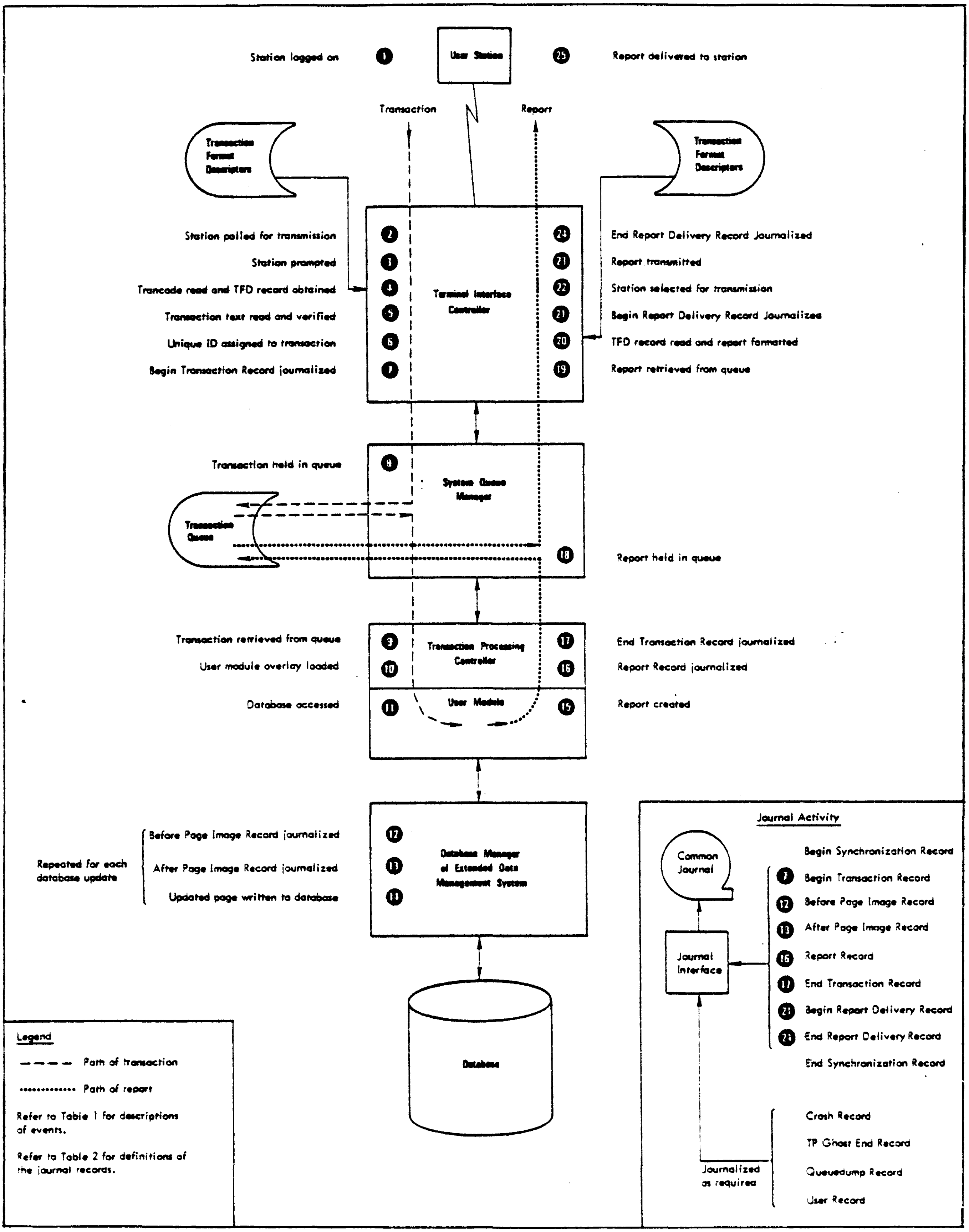
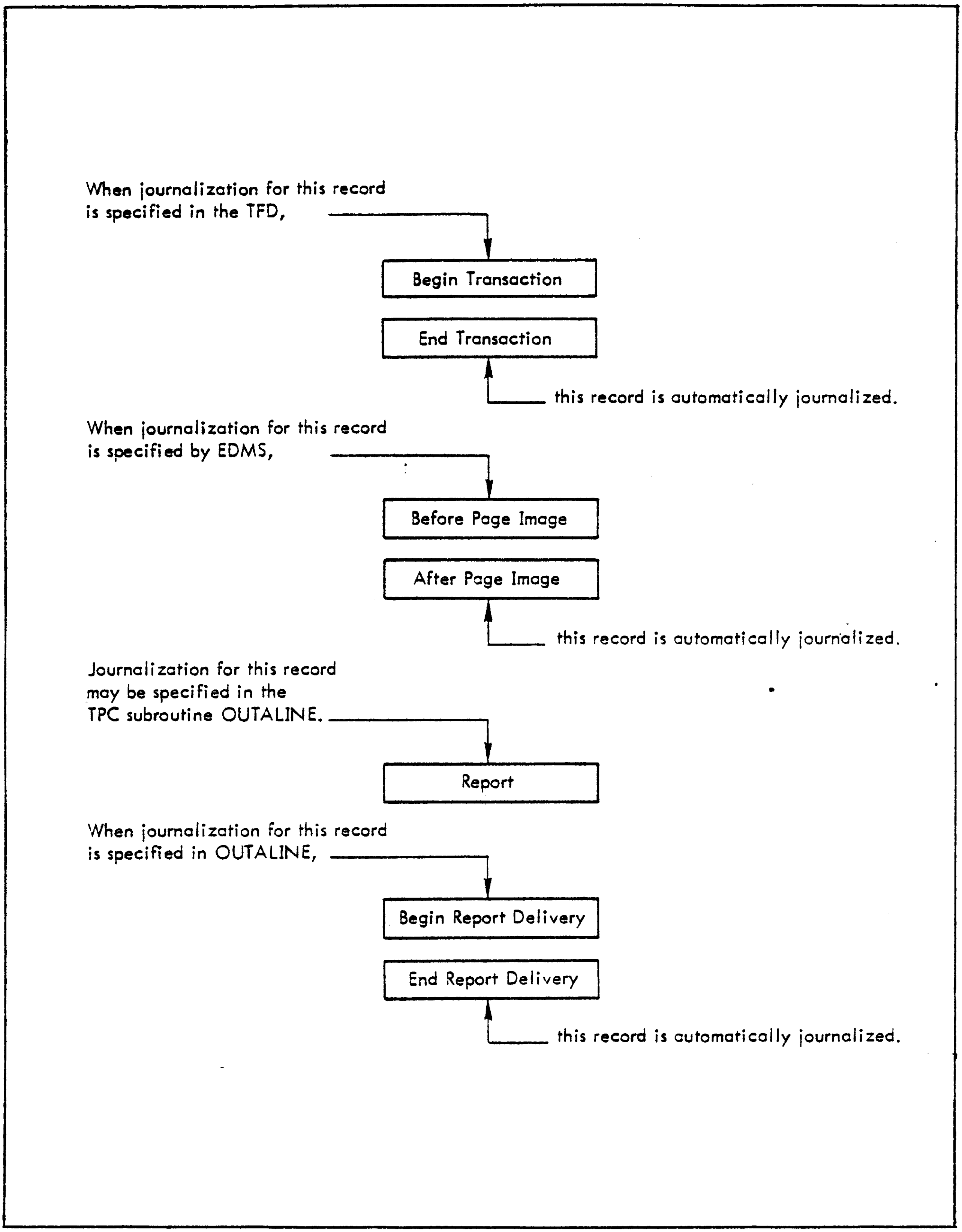

Station logged on
1
User Station
25
Report delivered to station
Transaction
Report
Transaction Format Descriptors
Transaction Format Descriptors
Station polled for transmission
Station prompted
Trancode read and TFD record obtained
Transaction text read and verified
Unique ID assigned to transaction
Begin Transaction Record journalized
2
3
4
5
6
7
Terminal Interface Controller
24
23
22
21
20
19
End Report Delivery Record Journalized
Report transmitted
Station selected for transmission
Begin Report Delivery Record Journalized
TFD record read and report formatted
Report retrieved from queue
Transaction held in queue
8
System Queue Manager
Transaction Queue
18
Report held in queue
Transaction retrieved from queue
User module overlay loaded
Database accessed
9
10
11
Transaction Processing Controller
User Module
17
16
15
End Transaction Record journalized
Report Record journalized
Report created
Repeated for each database update
Before Page Image Record journalized
After Page Image Record journalized
Updated page written to database
12
13
14
Database Manager of Extended Data Management System
Database
Legend
– – – – Path of transaction
·············· Path of report
Refer to Table 1 for descriptions of events.
Refer to Table 2 for definitions of the journal records.
Journal Activity
Common Journal
Journal Interface
Begin Synchronization Record
7 Begin Transaction Record
12 Before Page Image Record
13 After Page Image Record
16 Report Record
17 End Transaction Record
21 Begin Report Delivery Record
24 End Report Delivery Record
End Synchronization Record
Journalized as required
Crash Record
TP Ghost End Record
Queuedump Record
User Record

Figure 12-2. Chronological Sequence of Events

Table 12-1. Chronological Sequence of Events

| Event No. | Event |
|---|---|
| 1 | The Station user logs onto the system. |
| 2 | The Terminal Interface Controller polls the station for message transmission. |
| 3 | The Terminal Interface Controller prompts the station user, requesting a trancode. A trancode identifies a transaction type. |
| 4 | The Terminal Interface Controller receives the trancode from the user and finds the corresponding TFD. |
| 5 | The Terminal Interface Controller accepts the transaction text from the user and verifies it with the TFD. |
| 6 | The Terminal Interface Controller assigns a unique ID to the transaction in order to keep track of transactions as they flow through the system. |
| 7 | The Terminal Interface Controller journalizes the Begin Transaction record. |
| 8 | The Terminal Interface Controller passes the transaction to the System Queue Manager. In turn the System Queue Manager queues the transaction and posts an event indicating to the TPC that a transaction is available for processing. |
| 9 | The TPC retrieves the transaction from the System Queue Manager. |
| 10 | The TPC loads the user module that corresponds to the transaction type and passes control to the user module. |
| 11 | The user module typically accesses the database by calling library routines of the EDMS Database Manager. |
| 12 | The Database Manager journalizes the Before Page Image record. |
| 13 | The Database Manager journalizes the After Page Image record. |
| 14 | The Database Manager writes the modified page back to the database. |
| 15 | The user module creates the report or reports that may result from the transaction. These reports have the same ID assigned to the transaction. |
| 16 | The TPC journalizes one or more Report records, as required. |
| 17 | The TPC journalizes the End Transaction record. |
| 18 | The TPC passes a report to the System Queue Manager. In turn the System Queue Manager queues the report and posts an event indicating to the Terminal Interface Controller that a report is available for delivery. |
| 19 | The Terminal Interface Controller retrieves the report from the queue. |
| 20 | The Terminal Interface Controller finds the TFD that corresponds to the report type and formats the report according to the TFD specifications. |
| 21 | The Terminal Interface Controller journalizes the Begin Report Delivery record. |
| 22 | The Terminal Interface Controller selects the station for transmission. |
| 23 | The Terminal Interface Controller transmits the report to the station. |
| 24 | The Terminal Interface Controller journalizes the End Report Delivery record. |
| 25 | The report is printed or displayed at the station. |



When journalization for this record is specified in the TFD,

this record is automatically journalized.

When journalization for this record is specified by EDMS,

this record is automatically journalized.

Journalization for this record may be specified in the TPC subroutine OUTALINE.

When journalization for this record is specified in OUTALINE,

this record is automatically journalized.

Figure 12-3. Journalization Scheme

Table 12-2. Common Journal Records

| Record | Definition |
|---|---|
| Begin Synchronization | Indicates that the database area has been opened. EDMS journalizes this record when opening an area (CP-V random file) of the database. |
| Begin Transaction | Indicates that the transaction has entered the system. The Terminal Interface Controller journalizes this record before the transaction is queued. The TFD for the transaction type specifies whether or not transactions are to be journalized. |
| Before Page Image | Journalizes the database page before the database is modified. Journalization of this record is specified in EDMS. |
| After Page Image | Journalizes the database page after the database has been modified. When Before Page Image records are journalized, After Page Image records are automatically journalized. |
| Report | Journalizes the report contents. Report journalization is specified in the TPC subroutine OUTALINE. |
| End Transaction | Proves that the transaction has been processed and indicates whether it was successful or had failed. When Begin Transaction records are journalized, End Transaction records are automatically journalized. |
| Begin Report Delivery | Indicates that the Terminal Interface Controller is beginning to deliver the report or reports. Journalization of this record is specified in the TPC subroutine OUTALINE. |
| End Report Delivery | Indicates that the Terminal Interface Controller has completed delivery of the report. When Begin Report Delivery records are journalized, End Report Delivery records are automatically journalized. |
| End Synchronization | Indicates that the database area has been closed. |
| Crash | CP-V recovery automatically writes this record after a system failure when closing journals. |
| TP Ghost End | The TP Ghost automatically writes this record to journals before closing them. This occurs whenever the TP Ghost terminates. |
| Queuedump | This record is a snapshot of the transaction queue as it currently appears on disk. It is automatically written to the common journal when the TP Ghost opens the next tape in the series. When this record is journalized, together with its subsequent transaction records, reconstruction of the queue is expedited by limiting the number of journal tapes to be scanned. |
| User | User records may appear on the journal. They can be written with the TPC subroutine JOURNAL. |

Recovery handles several types of failures. It handles the abort of a single transaction, the abort of a system component, and a crash of the system itself. Recovery also handles the destruction of a critical file, such as the database or the transaction queue.

## ABORT OF A SINGLE TRANSACTION

If a user module employing EDMS aborts while modifying the database, the integrity of the database cannot be guaranteed. Should this condition occur, EDMS together with the TPC, rolls back the database - backs it up to reflect its contents before the transaction was processed - and reports the current transaction as failed to the TPC.

## ABORT OF A SYSTEM COMPONENT

Because the Terminal Interface Controller and the TPC execute as standard CP-V jobs, abort of either job stops transaction processing. Abort of the TP Ghost stops queuing and journalization. Should a system component abort, the operator may in effect first stop TP, without damage to CP-V, and then run standard recovery (discussed below).

## SYSTEM CRASH

Events associated with the CP-V monitor could also crash the system. In this case, CP-V recovery automatically checks all open files for incorrect linkages and inconsistencies. It records the cause of the crash, as well as other pertinent data for subsequent analysis. Standard recovery can then be run.

## DESTRUCTION OF CRITICAL FILES

The database or the transaction queue could be destroyed because of a hardware or a system malfunction or because of errors in user modules. The recovery processes can reconstruct the database and the transaction queue only for failures within TP and CP-V. Recovery from a malfunctioning user module, or restoration of the wrong database during system start-up, must be handled at the installation. Standard recovery provides for the inclusion of recovery processes at the installation.

## STANDARD RECOVERY

Standard recovery is initiated by the central operator as a series of batch jobs when a failure - other than the abort of a single transaction - occurs in CP-V or TP. (Standard recovery assumes that the abort of a single transaction has already been handled by the TPC/EDMS interface.) The phases of standard recovery are designed as separate processors rather than as part of the TP Ghost or as a single processor. This allows the installation to merge processors into standard recovery to handle special cases, and also provides restart points if a failure should occur during recovery. The recovery process is effectively checkpointed at the end of each phase.

Figure 12-4 is a functional diagram of recovery. CP-V recovery performs standard cleanup following a failure. The central operator than restarts the TP Ghost, which verifies the integrity of the transaction queue and sends recommendations to the operator. If the TP Ghost determines that the transaction queue must be reconstructed, then a processor named QREMAKE must be executed to reconstruct the transaction queue from the common journal. Standard recovery is then run to rollback the database, and to reset the status of "in progress" transactions and reports to allow reprocessing when the system has recovered.

Standard recovery consists of several phases. During one phase, for example, a processor named LISTQIP locates transactions and reports, if any, that were being processed when the failure occurred, and during another phase another procesor QPREP prepares the transaction queue for restart by handling the "in progress" transactions and reports located by LISTQIP. At each phase, the next phase to be run is determined, e.g., if no transactions or reports are in progress when the TP Ghost examines the transaction queue, then the execution of some phases can be omitted.

# ADDITIONAL FEATURES AND VARIATIONS IN SYSTEM USAGE

## PROTECTION AND CONTROL FEATURES

TP provides extensive protection. User modules are written in the traditional manner and protection is provided centrally within the system. Some of the protection and control features are

- Each user station is authorized by a log-on procedure in order to control access to TP components.

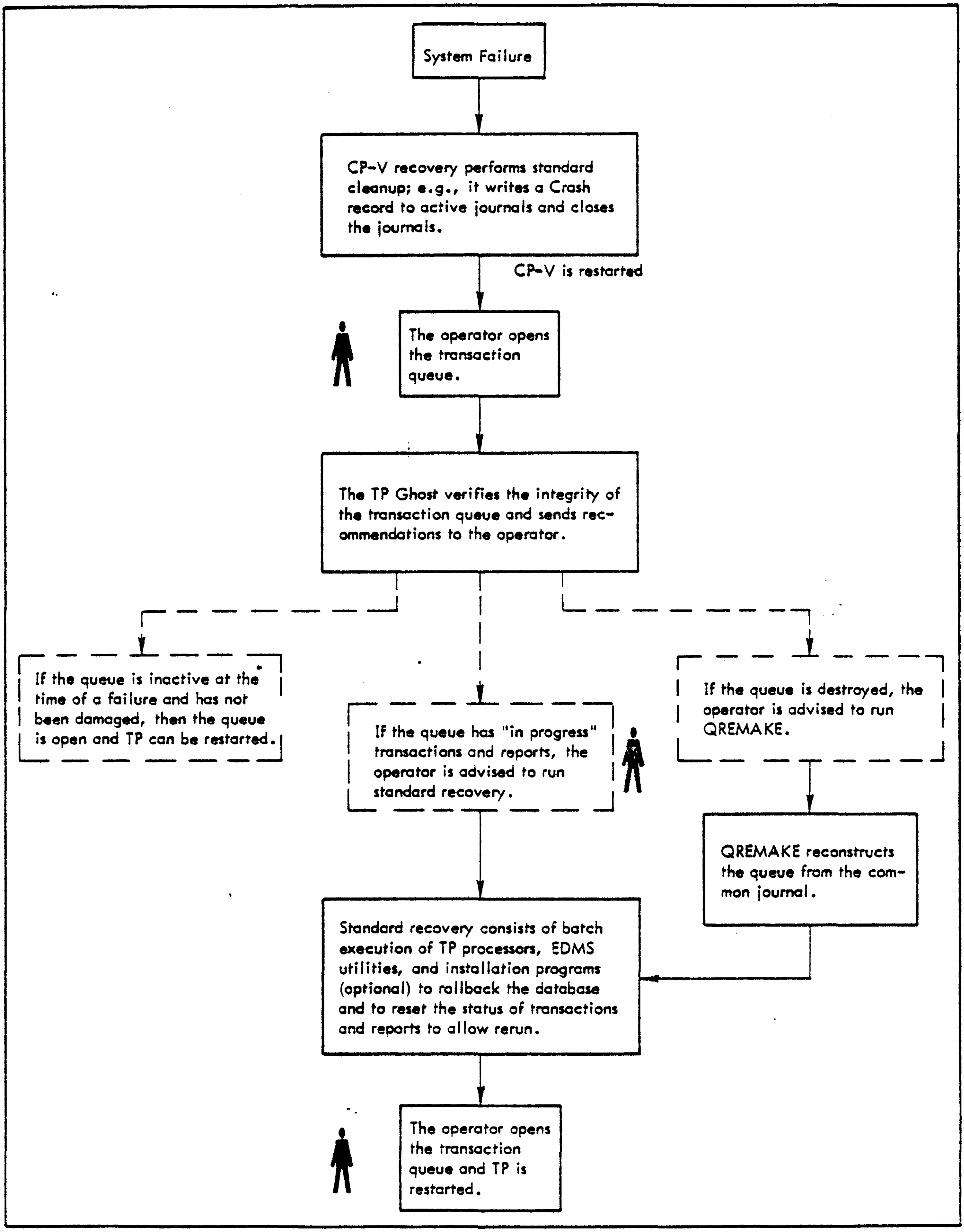

System Failure
CP-V recovery performs standard cleanup; e.g., it writes a Crash record to active journals and closes the journals.
CP-V is restarted
The operator opens the transaction queue.
The TP Ghost verifies the integrity of the transaction queue and sends recommendations to the operator.
If the queue is inactive at the time of a failure and has not been damaged, then the queue is open and TP can be restarted.
If the queue has "in progress" transactions and reports, the operator is advised to run standard recovery.
If the queue is destroyed, the operator is advised to run QREMAKE.
QREMAKE reconstructs the queue from the common journal.
Standard recovery consists of batch execution of TP processors, EDMS utilities, and installation programs (optional) to rollback the database and to reset the status of transactions and reports to allow rerun.
The operator opens the transaction queue and TP is restarted.

Figure 12-4. Functional Diagram of Recovery

- A transaction may be journalized. This ensures that it will not be "lost" - between its entry into the system and its processing - if an error occurs on the random access device used for queuing.

- When journalization is in effect, TP records the successful processing of transactions. This ensures that successful transactions will not be rerun should an error occur.

- EDMS may generate a record of the premodified database so that changes made by an aborted transaction may be undone or a damaged database may be restored.

- TP does not deliver reports until the transaction is successfully processed. Because reports and Begin and End Report Delivery records can be journalized, the loss of created - but undelivered - reports can be corrected.

## SYSTEM QUEUE MANAGER VARIATIONS

The System Queue Manager is designed with an interface such that other processors may be substituted for, or added to, the Terminal Interface Controller and the TPC. For example, the interface could be employed by user programs executing under CP-V time-sharing. It may be used as communication between two batch programs. It may also be used as a data collection medium where transactions are received at random intervals during the day, stored in the transaction queue, and processed during slow hours by a batch program similar to the TPC.

## SPAWNED TRANSACTIONS

One transaction may generate other transactions. Transactions are spawned in user modules by calling the TPC subroutine OUTATRAN, which is typically used to spawn one or more transactions on a low priority basis. Spawned transactions break down a unit of work in order to improve response times. In an inventory application, for example, each time a part is removed from stock, the relevant user module decrements the number-in-stock item for that part. When number-in-stock is reduced to the re-order point, a transaction is spawned to place an order for that part.

## OPERATIONAL CONSIDERATIONS

Some operational features to be considered are

1. A multiprogramming application may have more than one transaction processing load module, gaining the efficiency of that mode.

2. Under the direction of the system manager, the central operator may adjust the communications network by freeing and acquiring stations. The operator may also examine the status of transactions and control the distribution of reports.

3. TP does not affect the availability of the current CP-V batch and time-sharing services. These services can be used concurrently with transaction processing. Batch files and TP files are compatible.

4. TP provides a controlled interface between user modules and the EDMS database files accessed by those modules. It does not, however, provide this interface for any database other than an EDMS database. The user module communicates with those files using the CP-V file management system.

5. CP-V handles file contention problems.

6. CP-V enables the use of EDMS as a public library capable of concurrently serving the database access requirements of multiple on-line and batch users.

7. TP is fully compatible with, and operates in, the CP-V environment. All user programming interfaces involving EDMS, COBOL, FORTRAN, or Meta-Symbol are compatible with existing versions of those processors.

## TRANSACTION PROCESSING TERMINALS

Terminals supported include

- Xerox Model 7015 Keyboard Printer.

- Teletype Models 33, 35, 37, and 38.

- IBM 2741 Terminals.

- Tektronix Models 4010 and 4013.
- Datapoint 3300.
- IBM 3270: Display Stations 3277 (attached to 3271 Control Unit) and 3275.
- Terminals compatible with any of the above.

In addition, a class of message mode terminals is supported, including those that may be connected to bipoint and multipoint lines.

# CHAPTER 13. SYSTEM MANAGEMENT FACILITIES

The manager of a CP-V system may exercise considerable control over the system. This control is exercised through the use of the following facilities:

- System Generation
- User Authorization
- Use Accounting
- System Performance Control
- File Maintenance

## SYSTEM GENERATION

CP-V system generation is a multipass process by which the user can generate an operating system tailored to the requirements of a specific installation. Starting with a CP-V master system tape, the user can create a bootable system tape from which the generated operating system can be loaded into a target machine. The target machine can be any system having a hardware configuration compatible with CP-V and may have more or less core storage than the one used to generate the system tape.

The master tapes contain a bootable monitor, files of load modules (LMs) comprising the processors and other routines to be used during system generation, and a large number of element files (mostly ROMs) that constitute a data base for the system generation process. The user may patch the operating system as it is loaded into the machine from tape. When the monitor has been booted and the nonresident routines have been written to the disk, the CP-V system is fully operational.

The system generation process for the CP-V operationg system is performed by service processors. These processors operate as ordinary batch or on-line jobs to collect, compile, load, and write the modules required for a system. The service processors are as follows:

| Processor | Function |
|---|---|
| PCL | Selects from various sources the relevant modules required in the system generation process. |

| Processor | Function |
|---|---|
| PASS2 | Generates the required dynamic tables for the target resident monitor. |
| LOCCT and PASS3 | Stores and executes load card images (by calling the loader) to produce load modules (LMs) for the monitor and its processors. |
| DEF | Writes a monitor system tape that may be booted to bring up the target system. |

Control commands read by the PCL processor allow the user to select files from the data base of the master system tape, to substitute updated files for these (if necessary), and to add files to the resulting revised data base that is maintained in disk storage for use in later phases of the current system generation.

The PASS2 processor reads system generation control commands and generates disk files of load modules that establish operational labels, peripheral characteristics, logical device streams, allocatable system resources, and other installation dependent parameters that will be used during a later phase of the current system generation.

The object modules selected with PCL must be combined in load module form before a generated system tape can be written. Also, the tree structures for any overlays must be established. A tree table for each CP-V standard processor is present in the master release tape. However, tree tables for nonstandard processors must be created by the user through appropriate LOAD and TREE control commands.

After the user has created a tree table for an overlay structure, he has the option of using the LOCCT processor to generate a permanent LOCCT file containing the tree information so that this information need not be created anew during subsequent system generations.

If the generated system is to include CP-V standard modules or user-defined programs having associated LOCCT files of tree information, the PASS3 processor must be called to initiate the formation of load modules for them.

The PASS3 processor reads control commands specifying which LOCCT tables are to be used to define the load structure of CP-V standard modules or user-defined programs.

The user may specify that a given LOCCT table and associated object modules are to be deleted from disk storage after the component object modules have been loaded.

The first command read by PASS3 should specify the monitor's LOCCT table (e.g., M:MON), so that the monitor will be loaded first. Any items loaded will be biased to that bias contained in the LOCCT table for the item (this bias comes from the original LOAD control command used to generate the LOCCT table being used).

Items not specified in PASS3 control commands may be loaded via LOAD, OVERLAY, and TREE commands as in ordinary batch processing. (The monitor load module (M:MON) and RECOVER must be loaded with PASS3.)

When all desired object modules have been converted to load module form, the DEF processor must be called to write a tape containing the generated system.

The system tape generated by the DEF processor has the same general format as the master tape used in booting the CP-V system employed in the system generation process. The method of loading the generated system into the target machine is identical to that used in booting from the master tape.

To simplify the system generation process, standard monitor systems and standard processors are predefined in files on the CP-V release tapes. All LOCCTs and PASS2, PASS3, and DEF jobs are also included. These files are described in the release documentation. Rather than perform the entire system generation from scratch, a system manager can edit these files (using the Edit processor) to tailer the standard system generation to the needs of the installation.

## USER AUTHORIZATION

During log-on, four items are requested from the user: account, name, password, and extended accounting. (Password and extended accounting are optional.) These items are used to reference a log-on file that controls the entry of the job and, if the job is allowed, the type of usage and system privileges extended to the user.

The log-on file exists in the :SYS account under the name :USERS. It is composed of a series of records, one for each user who is authorized to log on. Most of these records are created by the system manager using the processor Super. The one exception is the :SYS account with the user name LBE. The first time there is an attempt to log on under account :SYS and user name LBE, a record for this account and name is automatically generated and placed in the :USERS file.

Records within the log-on file are keyed records with the key formed by the concatenation of account number and name of each valid user. Each record contains the identifying information, a password (which may be changed by the PASSWORD command) and other information that controls the system facilities granted to the user.

In addition, Super is used to create and maintain a file in the :SYS account called :PROCS. This file is keyed similarly to the :USERS file but does not necessarily contain a record for each user. The :PROCS file allows the system manager to restrict a user <u>to</u> an individually specified set of processors or to restrict an individual user <u>from</u> a specified set of processors. These restrictions may be individually controlled for the three modes of user access (on-line, batch, and ghost). The processors listed may be :SYS processors (both shared and unshared) or any executable load modules in any account.

Super is also used to create and maintain the :RBLOG file in the :SYS account which contains remote processing workstation authorizations. The records within this file contain information such as workstation name, type of remote terminal to be used at the workstation, maximum priority for jobs submitted from the workstation, and remote peripheral devices to be associated as part of the workstation.

## USE ACCOUNTING

A wide assortment of statistics accumulated during execution of each job is maintained in an accounting file by the system. Each installation may assign weighted charge values to each of the machine resources, and the system will use these to calculate and report costs accordingly. In addition, facilities are available to allow each installation to augment system accounting with routines unique to its particular needs.

Accounting statistics are gathered throughout the CP-V operating system. The first or initialization phase of gathering statistics involves the authorization of a user to address the system. This is set up by the system processor, SUPER, which maintains the log-on file, :USERS, and the log-on file for remote batch, :RBLOG. The log-on file information controls the user's access to the system resources and is static except for the updating of accumulated granule space. The second or accumulation phase is centered around the information acquired in the user's JIT, Job Information Table, and in his AMR, Assign-Merge Table (or Record). Each user receives an initialized JIT and Assign-Merge Table when the job or terminal session begins. The majority of accounting information is then accumulated in the user's JIT. The third or tabulation phase consists of tallying up the user's resource usage during log-off. An accounting record for the user is appended to the accounting file :ACCTLG, and the user is charged according to his established charge rates in the :RATE file.

The standard output of accounting information may take either one of two forms. One form consists of a summary of accounting information. The other form consists of the entire accounting record for the user.

For the on-line user, a summary of accounting information is sent to the terminal at the time the user logs off. The format of this information is

```
    CPU=m.mmm CON=h:mm INT=nn CHG=xxxx
```

where

m.mmmm is CPU time expressed in minutes and ten-thousandths of a minute.

h:mm is console time expressed in hours (h) and minutes (mm).

nn is the number of terminal interactions.

xxxx is total charge units for the on-line session.

The same information may be requested by the user during an on-line session by entry of the STATUS command, one of the commands of the Terminal Executive Language.

For the batch user, the entire accounting record is written. It is normally written to the line printer; however, at the batch user's request, it may be written to a file. The format of this printout is shown in Table 13-1.

Table 13-1. Accounting Printout for Batch Jobs

```
(Time and Date)

ELAPSED JOB TIME                                     hh:mm:ss

PARTITON NUMBER

CHARGE UNITS                                         xxxxxxxx

TOTAL CPU TIME                                         x.xxxx

     PROCESSOR EXECUTION TIME                          x.xxxx

     PROCESSOR SERVICE TIME                            x.xxxx

     USER EXECUTION TIME                               x.xxxx

     USER SERVICE TIME                                 x.xxxx

CARDS:    CARDS READ                                     xxxx

          CARDS PUNCHED                                  xxxx

PAGES:    PROCESSOR PAGES                                xxxx

          USER PAGES                                     xxxx

          DIAGNOSTIC PAGES                               xxxx

TAPES:    DRIVES ALLOCATED                                 xx

          TAPES MOUNTED                                    xx

PACKS:    SPINDLES ALLOCATED                               xx

          PACKS MOUNTED                                    xx

CORE:     PEAK CORE (PAGES)                               xxx

          PAGE * MINUTES                               xxxxxx

I/O:      OPERATIONS                                    xxxxx

          CALS                                         xxxxxx
```

Table 13-1. Accounting Printout for Batch Jobs (cont.)

```
FILE SPACE

     PEAK RAD TEMPORARY                                xxxx

     NET RAD PERMANENT                                 xxxx

     AVAILABLE RAD PERMANENT                           xxxx

     PEAK DISK TEMPORARY                               xxxx

     NET DISK PERMANENT                                xxxx

     AVAILABLE DISK PERMANENT                          xxxx

     NUMBER OF SWAPS                                   xxxx
                    :
     RESOURCES ALLOCATED
     CO = xx    9T = xxxx    7T = xxxx               (etc.)
```

## SYSTEM PERFORMANCE CONTROL

CP-V is a multiprogrammed, partition system that was designed to maximize utilization of the system's resources. Job throughput is dependent upon the efficiency with which system resources (i.e., core, tape drives, disk pack spindles, etc.) are utilized. A crude measure of efficiency is the percentage of time that each device and the CPU are busy for a given work load over a given period of time. Efficiency goes up and throughput increases when the resource utilization is greater for a particular work load and time sample. For a varying work load, however, high throughput will not always result from simultaneously high usage of all system resources.

Greater efficiency may be realized by overlapping I/O functions. One method of accomplishing this is to allow several jobs to reside in core concurrently, each receiving a time slice. (This concept is referred to as multiprogramming.) If the currently executing job issues an I/O call that causes physical I/O to occur, its quantum is ended and another job is scheduled and begins execution (i.e., it receives the usage of the CPU resource). Thus, two system resources, the I/O device called by the first job and the CPU, are now being utilized concurrently. It is easy to extrapolate from here to visualize several tape drives, a RAD, a disk pack, two line printers, and a card reader all operating simultaneously. The cost of achieving this overlap is, of course, more core since it is required for all processes whether I/O-bound or CPU-bound.

However, resource overlap will not occur if, say, three compute-bound jobs are scheduled for execution concurrently. Each job will, in turns monopolize the only resource all three need (the CPU) while other resources stand idle. This is why the Multi-Batch Scheduler (MBS) is needed. One of the main functions of MBS is to schedule jobs for concurrent execution so that they utilize as many resources as possible, and not to schedule jobs that will vie for a single resource, which would cause one or the other to occupy available core space (itself a resource) while waiting for a resource to be freed. Ideally, a multi-batch scheduler would schedule a compute-bound job with several I/O bound jobs and would let the compute-bound job take up the CPU slack while the others wait for I/O to complete.

Thus far, the discussion of batch system performance has approached the subject of resource optimization based on only one criterion - gross work accomplished per time unit. If the total system work done over, say, a twenty-four hour period were the only consideration, the discussion might stop here. However, all installations have unique user requirements and operational procedures, and diverse machine configurations. Consequently, there are certain additional criteria on which system performance must be judged. These criteria might be termed <u>operational considerations</u> and with each of them is associated a priority that is higher than the one assigned to raw throughput.

A hypothetical illustration of an operational consideration might be an installation that has a system configuration utilizing six tape drives. Experience at this installation has shown that when a set of jobs that uses all drives comes up for execution, it is all the operator can do to mount and dismount the required tapes and respond to the messages that appear on the operator's console. Also, it is known that between the hours of 3:00 and 4:00 p.m. on inordinate number of small listing jobs are submitted for processing. Those jobs normally occupy the operator's entire time in separating the output. Therefore, the installation manager may wish to block execution of either the job set requiring six tape drives or the listing jobs between the hours of 3:00 and 4:00 p.m. A more common situation would be one in which an installation must guarantee fast turnaround on jobs of short duration that use minimal resources while jobs of long duration or those that use tape drives and private disk packs must be given a smaller share of the CPU time until the fast turnaround jobs are run. Both of these examples illustrate an important principle that emerges as a consequence of tempering raw throughput with operational considerations - submitted batch jobs must have attributes defined in terms of necessary maximum resources to run the job. This is necessary so the system may identify those attributes, categorize the job, and schedule it to be run so as to satisfy the operational considerations while guaranteeing maximum throughput.

It is the responsibility of the user to specify the attributes of his job on the LIMIT command so that his job will not be scheduled for execution in the same manner as one requiring a greater slice of the system's resources.

The system manager is able to allocate the resources of his system to jobs with certain attributes by defining batch partitions under which diverse categories of jobs may run. A partition is a collection of ranges of job attributes. In some operating systems, a partition is defined as a fixed, addressable area of core in which jobs with certain attributes may run. Partitions in CP-V are not that type. No physical system resources such as core, spindles, or tape drives are permanently allocated to a partition. All jobs executing in the various partitions draw their physical resource requirements from a common pool without regard to the partition under which they qualified for execution except that the numeric limits that pertain to that partition will apply. Examples of attributes that comprise a partition are:

- Minimum and maximum job execution time
- Minimum and maximum amount of core
- Minimum and maximum number of disk pack spindles
- Minimum and maximum number of tape drives

All jobs entering CP-V for batch execution share the same input queue (often referred to as the 'batch job stream'). Jobs are selected from this queue for execution in the batch partitions.

Scheduling is performed in the following manner:

1. Available resources are determined.
2. The highest priority job requiring only available resources is selected.
3. The partition tables are searched for a partition that fits requested resources and is currently available.
4. If a partition is not available for the selected job, the next job is considered as in steps 2 and 3.

In summary, partition definitions are a primary factor in the job selection process. The system manager may direct the power of his system to the categories of jobs he so chooses by means of those definitions.

A maximum of 16 partitions may be defined for any system. It is recommended that 16 partition definitions be generated for all systems unless core memory is a serious consideration. This will provide a variety of job attribute classifications and those partitions in excess of the operational number may be locked from use through a processor (control) available to the system manager.

In a time-sharing/batch processing system, emphasis may be given to batch processing by opening up more partitions. However, it should be noted that CP-V is a queue-driven system and tasks are selected from prioritized queues without regard to the source of the request (i.e., on-line, batch, or remote batch). When there is a heavy on-line user load, as the number of batch partitions increases the number of compute-bound tasks increases and each receives a small franction of the CPU time. This means that batch jobs will be able to get more CPU time because of large quanta assigned to the batch partitions. This will not make a significant dent in on-line response time because interactive requests have a higher priority than compute bound jobs. More attention may be given to certain categories of batch jobs by increasing the number of partitions suitable for them.

CP-V has a comprehensive set of performance measurement and system control facilities. These facilities allow the system manager to determine how the system is performing and to adjust critical operational parameters to achieve better performance.

The three processors that provide these facilities are

1. Control
2. STATS
3. Summary

The Control processor provides a means of control over system performance. Commands of the control processor enable the system manager to display certain measurements and to "tune" the system as needed by setting new values for parameters that affect system performance. Control provides commands for

- Display of system parameters.
- Modification of system control parameters.
- Display and modification of partition definitions.

The STATS processor performs two functions. One function consists of displaying selected performance data in real-time. The other function consists of creating "snapshot" records of performance data for later processing by the Summary processor.

The Summary processor provides a global view of system performance by formatting and displaying the statistical data collected by STATS. The input data for Summary is the SNAPSHOT file created by STATS. The output listings are generally large and therefore must be output to a file or on the line printer.

The Summary processor allows the user to

1. Request a chronological listing of snapshot data for one or more display groups.
2. Specify a sort filter to remove undesired snapshots from the sample for subsequent reports.
3. Request filtered, sorted, and ordered listings of snapshot data for one or more display groups.
4. Request filtered, sorted, ordered, and averaged listings of snapshot data for one or more display groups.
5. Request means, minimums, maximums, and standard deviations for all display groups computed using the snapshots which pass the sort filter and a user specified intensity range. Correlation coefficients are included in this report that are estimates of the linear dependence between any pair of monitored variables.

## MAINTENANCE OF THE FILE SYSTEM

CP-V provides a variety of processors designed to meet the need for maintaining a reliable backup of the file system. A reliable backup of the file system is required for several reasons.

- The hardware may fail, either resulting in physical damage to the storage device or, more likely, presenting bad data to the software which may cause loss of or damage to the files.

- In the course of operation, demand for file space often exceeds availability, in which case it is necessary to move older files to secondary storage to make room for newer files.

- Privileged users can potentially damage the file system.

- Scheduled maintenance of the hardware sometimes requires use of the file devices by the Customer Engineer.

- Users make mistakes and delete files that they really wanted to keep.

Each file maintenance processor has a function for which it is uniquely suited. Together, the processors comprise a flexible mechanism designed to meet most file maintenance needs. Some typical file maintenance functions are:

- Saving individual files or groups of files on backup tape.

- Restoring individual files or groups of files from backup tape to the disk on which the file system is stored.

- Purging individual files or groups of files.

Most of the file maintenance processors can only be initiated by the central operator or by the file management portion of the monitor when space in the file system disk has reached a threshold low enough to warrant special action. However, the user may save a file or group of files onto the system backup tape at any time.

# CHAPTER 14. COMPUTER OPERATIONS

## SYSTEM START-UP AND INITIALIZATION

Several proceedures combine to cover the general subject of system start-up, initialization, and recovery from various levels of error situations. Each of the proceedures is tailored to restoration of the minimum amount of the system required to regain operation. Further recoveries proceed automatically whenever possible -- generally requiring no operator intervention.

When CP-V is in operation, secondary storage is divided into two logical parts:

- The file storage area (PFA) including its allocation tables.
- The system storage area (PSA) including the operating system, the shared processors, a copy of the file allocation tables, and the swapping area for user programs.

These two logical parts may reside on the same physical device but are usually separated onto separate devices: a moving arm disk for files and a fixed head RAD for system storage and swapping.

The recovery, restart, and initialization options include the following:

1. Automatic system recovery assuming valid contents of both PFA and PSA.
2. Operator directed recovery assuming valid contents of PFA and PSA.
3. Boot of a fresh system from tape after destruction of the PSA area, perhaps onto a different PSA device.
4. Boot of a new version of CP-V from tape without change of PFA. This and item three are referred to as "boot under the files".
5. Either three or four with an option to replace the system account files (:SYS) from the boot tape.

6. Complete boot from tape, recreating the file system anew and restoring file contents from copies saved on tape.

The last of these is the most comprehensive and thus is detailed below, even though in normal system use it is required very infrequently.

The operator begins system loading by mounting the current system tape on any 9-track tape unit and using the normal bootstrap procedure which is described in the operator's manual. After the initial portion of the tape has been read in, the following message is issued:

```
    ENTER ANY OF:
         I  =  TTY I/O
         P  =  LP OUTPUT
         F  =  TAPE FILES
         S  =  :SYS FILES
         T  =  TAPE PATCHES
         C  =  CARD PATCHES
         D  =  XDELTA
```

The response must end in new-line (NL), which is preceded by any combination of I, P, F, T, C, D, S, and N, or by nothing. If nothing, T is assumed.

I    enables the standard operator-system interaction during the boot (except the date/time request, which cannot be disabled). If I is not specified, certain messages and system requests are omitted from the load procedure.

P    enables printer output during the boot.

F    causes a new file system to be created. Its absence keeps the old file system (boot-under-the-files).

S    causes the files to be copied from the tape being read into the :SYS account without reinitializing the entire file system.

C and T    indicate that the patch deck(s) are to come from cards or tape, respectively. Either, neither, or both may be specified.

D    causes the XDELTA processor to be retained after the boot for system debugging purposes.

N    is meaningful only by itself and means "none of the above".

The system then allows the operator to

- Override the SYSGEN-defined values for the dedicated real-time memory pages (if any). ('I' must have been specified to do this.)

- Enter the date and time.

- Ensure that the card reader, line printer, and swapping disk are addressed correctly.

After this is completed, CP-V types its version, creation date, patching, and sense switch information. The operator may suppress some or all of this information, however, by depressing the BREAK key at the console an appropriate number of times.

CP-V then reads the patch deck. Any errors in the deck are displayed at the operator's console, along with the indication of the point-of-error within the incorrect field. The operator (or system programmer) must correct the patch card before initialization can continue. Patch card correction procedures are described in the CP-V/SP Reference Manual, 90 31 13.

If F was specified and the files are on a different tape, a :GENDCB card must be present in the patch deck. A mount message will be issued by CP-V. The operator must mount the tape and notify the system that the tape is ready. The tape is then copied to the file disk and the swapping disk is initialized.

During the time the tape is being copied to disk, the operator may adjust the number of on-line users allowed on the system by using the ON key-in (described later) or the OFF key-in if no on-line users are to be allowed.

After the tape has been read, the system is ready and terminal users receive the CP-V salutation and log-on request if their terminals are connected to the system.

To initiate the batch system, the operator places a deck of jobs to be run in a symbiont card reader and terminates the deck with a FIN control card. (The FIN control card contains !FIN in columns 1 through 4 and informs the system that the end of the deck of jobs has been reached.) The operator should then start the selected symbiont input device(s).

## JOB AND SYSTEM CONTROLS

The operator controls system operation through the use of console key-ins. These key-ins are listed in Table 14-1.

Table 14-1. Operator Key-ins

| Key-In | Function |
|---|---|
| ABORT | Abort user or job. |
| ANSMOUNT | Inform monitor that an ANS tape has been mounted. |
| ANSSCRATCH | Inform monitor that an ANS scratch tape has been mounted. |
| D | Enter date. |
| DATE | Enter date. |
| DELETE | Delete symbiont file from system. |
| DIAG | Authorize customer engineers to run diagnostics. |
| DISPLAY | Send system information to operator. |
| E | Error (terminate) job step - go to next job step. |
| ERROR | Error (terminate) job step - go to next job step. |
| ERSEND | Build a record in the system error log file. |
| FLUSH | Delete concurrent mode output being generated by a specified job for a specified device. |
| FORM | Change the form name on output files in the system. |
| GJOB | Initiate a ghost job. |
| HEADING | Provide message for on-line top-of-page heading or cancel previous heading. |

Table 14-1. Operator Key-Ins (cont.)

| Key-In | Function |
|---|---|
| INT | Transfer control to user's console interrupt routine. |
| MCSEND | Send message to a Xerox 560 Maintenance Control (Remote Assist Station). |
| MOUNT | Inform monitor that tape or pack is mounted. |
| OBOFF | Disallow entry of jobs to the batch stream from on-line terminals and processors. |
| OBON | Reallow on-line and processor entry of jobs to the batch stream after an OBOFF key-in. |
| OFF | Allow no more users to log on. |
| ON | Set maximum number of on-line users. |
| ONB | Set maximum number of batch users. |
| OUTPUT | Place all output streams of a job into concurrent output mode or release a device from the concurrent output mode. |
| OVER | Override the rejection of an output tape. (Applicable only for ANS tapes in the semi-protective mode.) |
| PRIORITY | Change user file or execution priority. |
| RBBDCST | Add message to the remote message file. |
| RBDISC | Disconnect a remote processing terminal. |
| RBLOG | Allow automatic log-on of a remote processing terminal. |
| RBS | Allow connection of remote processing terminal. |
| RBSEND | Send a message to a remote terminal. |
| RBSWITCH | Switch output files from one workstation to another. |

Table 14-1. Operator Key-ins (cont.)

| Key-In | Function |
|---|---|
| RBX | Disconnect (and disallow connection of) one or all remote processing terminals. |
| READ | Inform the monitor that a tape without a write ring (for which the user specified that both reading and writing would be done) will be read only. |
| REQUEST | Prepare to dismount tape from unit ndd or request the tape type of a specified resource. |
| S | Search for input symbiont files to run. |
| SCPU | Start the specified secondary CPU. |
| SCRATCH | Use the specified tape as a scratch unit. |
| SEND | Issue message to a specific on-line user or to all on-line users. |
| SS | Start symbiont card reader. |
| START | Search for input symbiont files to run. |
| S | Initiate symbiont action. |
| T | Enter time. |
| TIME | Enter time. |
| X | Abort user or job. |
| XCPU | Stop specified secondary CPU. |
| ZAP | Abort all users and save the symbiont pointers for restart. |
| device,action | Initiate action indicated on the specified device (in response to a previous device message). |

## REMOVABLE STORAGE INITIALIZATION

VOLINIT provides for the initialization of public and private disk packs. It is used to establish serial numbers and ownership, to write headers and other system information in selected areas of the volumes, and to test the surface of the disks and select alternate tracks to be used in place of flawed tracks. There are two versions of VOLINIT. One is a CP-V processor which runs in the batch, on-line, or ghost mode under normal CP-V operation. The other version is a stand-alone processor which runs on the computer when CP-V is not in control.

The Label processor initializes ANS tapes by writing ANS formatted labels. (CP-V labeled tapes are not initialized; they are labeled as part of the I/O procedures.) A secondary purpose of Label is to label any unlabeled tape to be used in a protected or semi-protective system. In the ANS protective mode, all ANS tapes must be prelabeled by Label. In the semi-protective mode, ANS tapes may be prelabeled by Label or may be given ANS labels as the result of an operator key-in.

The Label processor performs the following three functions:

1. It initializes ANS tapes by writing standard expired ANS labels.

2. It creates "unlabeled" tapes to be used as scratch tapes. These tapes contain three dummy records and two tape marks which facilitate using new tapes as scratch tapes. (The tapes will no longer be degaussed and therefore will not "run away" when AVRed.)

3. It prints the contents of the header and trailer labels of input labeled tapes or the first 80 bytes of each block if the tape is not a labeled tape.

## PERIPHERAL DEVICE ERROR PROCEDURES

If the monitor encounters an abnormal condition during an I/O operation, it will send a message (Table 14-2) to the operator. These device error messages are generated both for errors that are irrecoverable and for errors that are recoverable with

Table 14-2. Device Error Messages and Operator Action

| Message | Operator Action |
|---|---|
| device MANUAL | Ready the device. |
| device WRITE PROTECT | Error (E) or remove write-protect and retry (R). |
| device TIMED OUT | Retry (R) or error (E). Time-out values are measured in ticks of a 5-second clock.<br>1. Tape rewind and space file - 50 ticks.<br>2. Operator terminal input - 100 ticks.<br>3. All others - 2 ticks. |
| device ERROR | Retry (R), continue (C), or error (E) if card reader of line printer; otherwise the error is irrecoverable and no operator action is needed or possible. |
| device NOT OPERATIONAL | Device busy, not recognized, or I/O not accepted. Correct the condition, usually dial tape unit or turn power on, and error (E) or retry (R). |

operator assistance. The operator may respond with a device key-in

```
device,action
```

where 'device' specifies the device and the 'action' is one of the following:

C continue as is.

E continue but inform the program of the error.

R retry I/O operation, possibly after correcting the problem (e.g., by moving the error card back to the read station).

If a required device is in manual status, the following message is typed every 20 seconds:

```
    device MANUAL
```

In all other cases, if an operator action is required and none is received, the following message is issued:

```
    device PLEASE RESPOND
```

A special form of message is issued for read, write, or write-check errors occurring on the swapping disk. A message is issued each time an initial or retry error occurs; multiple messages are issued if several types of errors are indicated on one operation. An accumulated count of retries within any one recovery attempt is given in the message. The format is:

```
    n op error
```

where

n    is the number of retries (0 for initial occurrence of the error).

op    is RD(read), WRT(write), or WCK(write-check).

error    is one of the following:

```
                IOP CONTROL ERR
                IOP MEMORY ERR
                MEMORY ADR ERR
                XMISS MEM ERR
                XMISS DATA ERR
```

In addition to logging errors on the operator's console, the system also maintains a system error log file called ERRFILE. This file contains a log of system and peripheral device failures that were corrected, that were irrecoverable, or that required operator assistance for recovery. (ERRFILE is described in greater detail in Chapter 16.)

## LETTING THE SYSTEM RUN ITSELF

An important feature of the CP-V system is that the computer operator may leave the system alone and let it run itself. This allows an installation to have selected periods of time (for example, grave yard shift) to run time-consuming batch jobs which require no peripheral device action on the part of the operator.

When allowing the system to run itself, the operator usually turns the line printer off. Thus, there will be no concern about a possible line printer failure in the operator's absence. Printer output will collect in the symbiont. When the operator returns, the line printer can be turned on and the collected output will be printed.

If some on-line or batch job happens to request an operator action such as the mounting of a tape, only that one job will be affected. The system continues operation in a normal manner.

# CHAPTER 15. RECOVERY

CP-V attempts to make the system available as much as possible with minimal loss of data when problems occur. To this end, a recovery package is available which takes actions based on the seriousness of any problem which occurs. CP-V accomplishes the recovery completely automatically, not requiring operator intervention of any kind.

The various modules of CP-V have code embedded in them to check the consistency of the resident operating system tables and the important user context. If an inconsistency is detected, or if a hardware error is reported which is judged to have compromised the integrity of the resident operating system, recovery is initiated and one of three actions is taken.

1. If the damage is judged to be isolated to the context of a single user, a procedure called Single User Abort is performed. This involves writing the contents of main memory to secondary storage, writing out updated file buffers for the user, and eliminating the user job (i.e., releasing his main memory and swap storage and removing all records of him from the user tables). The system is then allowed to proceed for all other users. Normal operation of the system is interrupted for less than five seconds.

2. If the damage is not isolated to the context of a single user but certain key system tables (Current File Use Tables, Allocation Tables, Job Pending Tables, etc.) are judged to be intact, a procedure called Normal Recovery is performed. The memory image is written to secondary storage. The context for each user (in memory or swapped out) is then examined. All open files are closed with default options (OUT files are released, etc.). Partial output symbiont files are packed and put into the output queue. Remaining input for batch jobs which are partially completed is discarded unless the user has specified the RERUN option in his job deck, in which case the job is put back into the job queue. The accounting information is saved along with all temporary file names. After all users have been processed in this manner, the remaining Current File Use Tables are examined. If any open INOUT files remain, the names are retained. Then the resident operating system is brought in from the system swapping device. Before resuming normal operation, accounting records are written and all temporary files are deleted. Previously recorded INOUT

files are copied over themselves to eliminate potential inconsistencies. At this point, normal system operation proceeds. Terminal users must log on again. This process requires one-half to five minutes depending on the number of users and other factors.

3. If the key system tables are damaged, a procedure called Extended Recovery is performed. The memory image is written to secondary storage. Each individual file in the system is then examined for space allocation information. The allocation tables are rebuilt and dual allocations (i.e., situations in which more than one file is trying to use the same space in the file system) are noted. When this process is complete, the system is reinitiated. The Extended Recovery procedure requires an amount of time which is proportional to the size of the file system (anywhere from 15 minutes to 5 hours).

After any of the three types of recovery has been performed, the monitor dump analysis program (ANLZ) is initiated to aid in determining the cause of the problem. The output produced by ANLZ consists of formatted displays of monitor and user tables and the contents of the registers existing at the time of the problem.

## SYSTEM ERROR LOG FILE

All hardware malfunctions and some software problems occurring during system operation, whether recovered or not, are recorded in a special disk storage file. This file is periodically copied into a standard file (ERRFILE) by a ghost program (ERR:FIL) which is initiated automatically for that purpose.

ERRFILE may be listed and summarized by the Error Log Listing processor (ELLA). ERRFILE is also available for on-line preventive maintenance of the system and for diagnosis and prediction of hardware malfunctions.

A sample of the types of conditions that are recorded in ERRFILE is given in the list below.

- An error was detected during memory access by either the CPU or an IOP.
- Execution of an SIO (Start I/O) instruction failed.
- An I/O interrupt did not occur within a specified time period in response to an I/O instruction (device timed-out).
- An interrupt, other than an attention interrupt, was received from a device for which no I/O operation had been started by the system.
- An error occurred during an I/O operation.
- System initialization or system recovery occurred.
- The system detected an inconsistency in the file system.
- A power-on trap occurred.
- A granule being released contained an invalid disk address or had already been released.
- An error occurred during the transmission of data to or from a remote processing workstation.
- A record was entered by the operator to describe unusual conditions surrounding a particular error.

- A memory parity error occurred.
- A resource was partitioned out of the system by the operator.
- A partitioned resource was returned to the system by the operator.

Not all ERRFILE records are the result of error conditions. For example, a time stamp record is entered once each hour on the hour and an I/O activity count is recorded each hour and at recovery.

The Error Log Listing processor (ELLA) provides an efficient tool for listing and sorting the error log file, ERRFILE. ELLA output furnishes a meaningful and comprehensive diagnostic evaluation of the system and its peripherals, aiding in the early detection of product failures and thus increasing the reliability, maintainability, and availability of the system.

The set of ELLA commands allows the user to first specify the kinds of errors in which he is interested, and then request a listing of those kinds. Four types of listings are available:

- A chronological listing of error log entries.
- A sorted listing of error log entries.
- A summary of error log entries by category.
- A summary of error log entries in graphic form.

## ON-LINE PERIPHERAL DIAGNOSTIC FACILITIES

Within the system, diagnostics are provided that may be used from either local or remote terminals to analyze and repair card readers, card punches, line printers, magnetic tapes, RADs, and disk packs. These run during system operation without disturbing on-line users or batch job throughput (except, of course, for jobs requiring the down device). Full direct access to the device is provided, and all hardware status information for the read or write operation is returned to the diagnostic. The diagnostics provide

- Functional tests for peripheral devices that isolate hardware problems to the lowest possible level.
- Exercisers that verify that the peripherals are operating correctly.

- Preventive maintenance tests that reduce the amount of time that peripherals are down for repair.

These tests and exercisers may be run at an on-line terminal while the CP-V system is in normal operation.

# APPENDIX A. CP-V PROCESSORS

## STANDARD CP-V PROCESSORS

The processors that are available for use with CP-V are briefly described in this appendix. References are provided for the reader who wants a detailed description. The following abbreviations are used in the references:

BP - batch processing

LN - language

OPS - operations

RP - remote processing

RT - real-time

SM - system management

SP - system programming

TP - transaction processing

TS - time-sharing

UT - utilities

## COMMAND PROCESSORS

There are four command processors: LOGON/LOGOFF, EASY, TEL, and CCI. The first of these processors is available to on-line and batch users, the second and third are available to on-line users only, and the last is available to batch users only.

### LOGON/LOGOFF

LOGON admits on-line users to the system and connects the user's terminal either to TEL or to an alternative processor, such as BASIC, that has been selected by the user. LOGOFF disconnects a user from the system and does the final cleanup and accounting.

## EASY

EASY is a shared processor that enables the user to create, edit, execute, save, and delete program files written in BASIC or FORTRAN. EASY also allows the user to create and manipulate EBCDIC data files. Although intended primarily for Teletype operations, EASY can be used with any type of on-line terminal supported by the system. (Reference: EASY/LN, OPS Reference Manual, 90 18 73.)

## TERMINAL EXECUTIVE LANGUAGE

The Terminal Executive Language (TEL) is the principal terminal language for CP-V. Most activities associated with FORTRAN, COBOL and assembly language programming can be carried out directly in TEL. These activities include such major operations as composing programs and other bodies of text, compiling and assembling programs, linking object programs, initiating execution, and debugging programs. They also include such minor operations as saving and restoring core images of programs for which execution was interrupted, determining program status, and setting simulated tab stops. (Reference: CP-V/TS Reference Manual, 90 09 07.)

## CONTROL COMMAND INTERPRETER

The Control Command Interpreter is the batch counterpart of TEL. It provides the batch user with control over the processing of batch programs just as TEL provides on-line users with control over the processing of on-line programs. (Reference: CP-V/BP Reference Manual, 90 17 64.)

# LANGUAGE PROCESSORS

Language processors translate high-level source code into machine object code. Seven processors of special importance are described below. All of these can be used in both on-line and batch mode.

## ANS FORTRAN

The ANS FORTRAN compiler is compatible with most features of the forthcoming (new) ANS Standard FORTRAN language which includes many extensions to the 1966 ANS FORTRAN Standard Language. It is operable under CP-V as a shared processor, offering services to both the batch user and the on-line user. The user may request, as an operation, that the compiler produce either ROM output or program execution (LOAD and GO).

Advantageous features of the ANS FORTRAN compiler are

- o Compiler speed on the order of 2K-3K lines per minute.
- o Compressed input/output capability.
- o Addition of INCLUDE (system) capability.
- o Conversational characteristics for time-sharing.
- o New ANS FORTRAN compatibility.
    - o CHARACTER variables.
    - o Expanded READ/WRITE capabilities.
    - o OPEN and CLOSE statements.

(References: ANS FORTRAN/LN Reference Manual, 90 32 00, and ANS FORTRAN/OPS Reference Manual, 90 32 01.)

## META-SYMBOL

Meta-Symbol is a procedure-oriented macro assembler. It has services that are available only in sophisticated macro assemblers and a number of special features that permit the user to exercise dynamic control over the parametric environment of assembly. It provides users with a highly flexible language with which to make full use of the available hardware capabilities.

Meta-Symbol may be used in either batch or on-line mode. When used in on-line mode, the assembler allows programs to be assembled and executed on-line but does not allow conversational interaction.

One of the many Meta-Symbol features is a highly flexible list definition and manipulation capability. In Meta-Symbol, lists and list elements may be conveniently redefined, thus changing the value of a given element.

Another Meta-Symbol feature is the macro capability. Xerox uses the term "procedure" to emphasize the highly sophisticated and flexible nature of its macro capability. Procedures are assembly-time subroutines and provide the user with an extensive function capability. Procedure definition, references, and recursions may be nested up to 32 levels.

Meta-Symbol has an extensive set of operators to facilitate the use of logical and arithmetic expressions. These operators facilitate the parametric coding capabilities available with Meta-Symbol (parameteric programming allows for dynamic specification of both "if" and "how" a given statement or set of statements is to be assembled).

Meta-Symbol users are provided with an extensive set of directives. These directives, which are commands intrinsic to the assembly, fall into three classes:

1. Directives that involve manipulation of symbols and are not conditionally executed.

2. Directives that allow parameteric programming.

3. Directives that do not allow parameteric programming.

A number of intrinsic functions are also included in Meta-Symbol. These give the user the ability to obtain information on both the structure and content of an assembly time construct. For example, the user can acquire information on the length of a certain list. He can inquire about a specific symbol and whether it occurs in a procedure reference. (Reference: Meta-Symbol/LN,OPS Reference Manual, 90 09 52.)

AP

Assembly Program (AP) is a four-phase assembler that reads source language programs and converts them to object language programs. AP outputs the object language program, an assembly listing, and a cross reference (or concordance listing). AP is available in both the on-line and batch modes.

The following list summarizes AP's more important features for the programmer:

- Self-defining constants that facilitate use of hexadecimal, decimal, octal, floating-point, scaled fixed-point, and text string values.
- The facility for writing large programs in segments or modules. The assembler will provide information necessary for the loader to complete the linkage between modules when they are loaded into memory.
- The label, command, and argument fields may contain both arithmetic and logical expressions, using constant or variable quantities.
- Full use of lists and subscripted elements is provided.
- The DO, DO1, and GOTO directives allow selective generation of areas of code, with parametric constants or expressions evaluated at assembly time.
- Command procedures allow the capability of generating many units of code for a given procedure call line.
- Function procedures return values to the procedure call line. They also provide the capability of generating many units of code for a given procedure call line.
- Individual parameters on a procedure call line can be tested both arithmetically and logically.
- Procedures may call other procedures, and may call procedures recursively.

(Reference: Assembly Program Reference Manual, 90 30 00.)

## BASIC

BASIC is a compiler and programming language based on Dartmouth BASIC. It is, by design, easy to teach, learn, and use. It allows individuals with little or no programming experience to create, debug, and execute programs via an on-line terminal. Programs are usually small to medium size applications of a computational nature.

BASIC is used primarily for on-line program development and execution, or on-line development and batch execution. In addition, programs may be developed and executed in batch mode.

BASIC provides two user modes of operation. The editing mode is used for creating and modifying programs. The compilation/ execution mode is used for running completed programs. This arrangement simplifies and speeds up the program development cycle.

Statements may be entered via a terminal and immediately executed. The principal benefit of direct execution is on-line development of programs and short simple computations. During execution, programs may be investigated for loop detection, snapshots of variables may be obtained, values of variables may be changed, flow of execution may be changed, flow of execution may be rerouted, and so on. This unique capability allows an on-line terminal to be used as a "super" desk calculator.

At compile and execute time, the user may specify if an array dimension check is to be made. In the safe mode, statements are checked to verify that they do not reference an array beyond its dimensions. In the fast mode, this time consuming check is not made. Thus, the safe mode could be used during checkout, and the fast mode could be used to speed up execution when the program reaches the production stage.

BASIC provides an image statement that uses a "picture" of the desired output format to perform editing. It also has TAB capability and a precision option to indicate the number of significant digits (6 or 16) to be printed.

An easy-to-use feature allows the user to read, write, and compare variable alphanumeric data. This is particularly important for conversational input processing.

Chaining permits one BASIC program to call upon another for compilation and execution without user intervention. Thus, programs that would exceed user core space may be segmented, and overlay techniques may be employed via the chaining facility. (Reference: BASIC/Reference Manual, 90 15 46.)

## ANS COBOL

The ANS COBOL compiler is a powerful and convenient programming language facility for the implementation of business or commercial applications. The language specification fully conforms to the proposed ANSI standard for the various functional processing modules. Only those language elements that cause ambiguities or are seldom used have been deleted. The compiler's design takes full advantage of the machine's unique hardware features, resulting in rapid compilation of source code, rapid execution of the resulting object code, and the generation of compact programs.

The result is a highly efficient programming system requiring a minimum amount of storage.

ANS COBOL contains many facilities that are either not found in other systems or, if available, are provided only at greater cost in terms of equipment required. Some of the facilities that provide more flexibility and ease of use in program development include

1. Implementation of table handling mode.
2. Sort/merge linkage.
3. Sequential access.
4. Random access linkage.
5. Segmentation.
6. Report writer.
7. Library utilization.
8. Calling sequence for FORTRAN, Meta-Symbol, etc.
9. Packed decimal as well as floating-point arithmetic formats.
10. Data name series options for ADD, SUBTRACT, MULTIPLY, DIVIDE, and COMPUTE verbs.

The system provides the user with a comprehensive set of aids to minimize the time required to print "bug-free" programs in the form of listings. These listings include

1. The source language input to the compiler with interspersed English language diagnostic messages.
2. An optional listing of the relocatable binary output, printed in line number sequence identical to the source language listing.

3. A cross-reference listing, indicating by line number where each data name or paragraph name is defined in the COBOL program and where each reference is located.

In addition, at run time, the user may use TRACE and EXHIBIT to follow execution of the procedure division.

The compiler is designed to take full advantage of high-speed, random access secondary storage (e.g., RAD storage). This feature means faster job execution because of minimized I/O delays, and smaller core memory requirements because of rapid overlay service. (Reference: ANS COBOL/LN Reference Manual, 90 15 00.)

## APL

APL is an acronym for A Programming Language, the language invented by Kenneth Iverson. It is an interpretive, problem-solving language. As an interpretive language, APL does not wait until a program is completed to compile it into object code and execute it; instead, APL interprets each line of input as it is entered to produce code that is immediately executed. As a problem-solving language, APL requires minimal computer programming knowledge; a problem is entered into the computer and an answer is received, all in the APL language.

Because APL is powerful, concise, easy to learn, and easy to use, it is widely used by universities, engineers, and statisticians. It also has features that make it attractive for business applications where user interaction and rapid feedback are key issues. One of APL's major strengths is its ability to manipulate vectors and multidimensional arrays as easily as it does scalar values. For example, a matrix addition that might require a number of statements and several loops in other languages can be accomplished as A+B in APL. This type of simplification exemplifies APL's concise power. (Reference: APL/LN, OPS Reference Manual, 90 19 31.)

## RPG

RPG (Report Program Generator) is a convenient means of preparing reports from information available in computer-readable forms, such as punched cards, magnetic tape, and magnetic disks. In addition, it is a means of establishing and updating files of information, usually in conjunction with preparation of reports.

RPG provides its capabilities through generation (compilation) of object programs, each of which is tailored to produce a different set of reporting results and/or file processing desired by the user. The RPG object programs are capable of accepting input data, retrieving data from existing files, performing calculations, changing formats of data, updating existing files, creating new files, comparing data values to one another and to specified constants to determine appropriate handling, using user-defined processing subroutines, using system library subroutines, and printing reports derived from the input and file data.

RPG has several advantages over the more traditional method of writing object programs in a symbolic programming language. The RPG language is oriented toward the user's problem, describing reporting requirements, rather than toward the mechanics and manipulations of computer usage. The language and specification techniques are easily learned. A user can become proficient in RPG after writing only a few programs, whereas an equal facility in symbolic programming would require considerable experience. (Reference: RPG/Reference Manual, 90 19 99.)

## EXECUTION CONTROL PROCESSORS

Processors in this group control the execution of object programs. Delta and COBOL On-Line Debugger can be used in on-line mode only. Load can be used in batch mode only. Link, LYNX, and FDP can be used in either batch or on-line mode.

### LOAD

Load is a two-pass overlay loader. The first pass processes

1. All relocatable object modules (ROMs).

2. Protection types and sizes for control and dummy sections of the ROMs.

3. Expressions for definitions and references (primary, secondary, and forward references).

The second pass forms the actual core image and its relocation dictionary. (Reference: CP-V/BP Reference Manual, 90 17 64.)

## LYNX

LYNX is a load processor that is available in both the on-line and batch modes. LYNX has most of the capabilities of the overlay loader and also provides the same control over internal and global symbol table construction which is available in the Link loader. LYNX may be veiwed as a preprocessor for the overlay loader. After it analyzes the user's commands, it constructs a table of loader control information which it then passes to the overlay loader. It is the overlay loader which actually performs the loading process. (Reference: CP-V/TS Reference Manual, 90 09 07, and CP-V/BP Reference Manual, 90 17 64.)

## LINK

Link is a one-pass linking loader that constructs a single entity called a load module, which is an executable program formed from relocatable object modules (ROMs). Link is now provided with CP-V only for compatability with previous versions of the system. Is is recommended that the Load or LYNX loader be used instead. (Reference: CP-V/TS Reference Manual, 90 09 07, and CP-V/BP Reference Manual, 90 17 64.)

## DELTA

Delta is designed to aid in the debugging of programs at the assembly-language or machine-language levels. It operates on object programs and tables of internal and global symbols used by the programs but does not require that the tables be at hand. With or without the symbol tables, Delta recognizes computer instruction mnemonic codes and can assemble machine-language programs on an instruction-by-instruction basis. The main purpose of Delta, however, is to facilitate the activities of debugging by

1. Examining, inserting, and modifying such program elements as instructions, numeric values, and coded information (i.e., data in all its representations and formats).

2. Controlling execution, including the insertion of break-points into a program and requests for breaks on changes in elements of data.

3. Tracing execution by displaying information at designated points in a program.

4. Searching programs and data for specific elements and subelements.

Although Delta is specifically tailored to machine language programs, it may be used to debug any program. Delta is designed and interfaced to the system in such a way that it may be called in to aid debugging at any time, even after a program has been loaded and execution has begun. (Reference: CP-V/TS Reference Manual, 90 09 07.)

## FORTRAN DEBUG PACKAGE

The FORTRAN Debug Package (FDP) is made up of special library routines that are called by ANS FORTRAN object programs compiled in the debug mode. These routines interact with the program to detect, diagnose, and in many cases, repair program errors.

The debugger can be used in batch and on-line modes. An extensive set of debugging commands are available in both cases. In batch operation, the debugging commands are included in the source input and are used by the debugger during execution of the program. In on-line operations, the debugging commands are entered through the terminal keyboard when requested by the debugger. Such requests are made when execution starts, stops, or restarts. The debugger normally has control of such stops.

In addition to the debugging commands, the debugger has a few automatic debugging features. One of these features is the automatic comparison of standard calling and receiving sequence arguments for type compatibility. When applicable, the number of arguments in the standard calling sequence is checked for equality with the receiving sequence. These calling and receiving arguments are also tested for protection conflicts. Another automatic feature is the testing of subprogram dummy storage instructions to determine if they violate the protection of the calling argument. (Reference: FDP/Reference Manual, 90 16 77.)

## COBOL ON-LINE DEBUGGER

The COBOL On-Line Debugger is designed to be used with ANS COBOL. The debugger is a special COBOL run-time library routine that is called by programs compiled in the TEST mode. This routine allows the programmer to monitor and control both the execution of his program and the contents of data-items during on-line execution. The debugger also allows the COBOL source program to be examined and modified.

The debugger can only be used during on-line execution; however, programs that have been compiled for use with the debugger may be run in the batch mode. This is not recommended, though, because of the increased program size when the TEST mode is specified. (Reference: ANS COBOL On-line Debugger Reference Manual, 90 30 60.)

## SERVICE PROCESSORS

The processors in this group perform general service functions required for running and using the CP-V system.

### EDIT

The Edit processor is a line-at-a-time context editor for on-line creation, modification, and handling of programs and other bodies of information. All Edit data is stored on disk storage in a keyed file structure of sequence numbered, varied length records. This structure permits Edit to directly access each line record of data.

Edit functions are controlled through single line commands supplied by the user. The command language permits insertion, deletion, reordering, and replacement of lines or groups of lines of text. It also permits selective printing, renumbering records, and context editing operations of matching, moving, and substituting line-by-line within a specified range of text lines. File maintenance commands are also provided to allow the user to build, copy, merge, and delete whole files. (Reference: CP-V/TS Reference Manual, 90 09 07.)

### PERIPHERAL CONVERSION LANGUAGE

The Peripheral Conversion Language (PCL) is a utility subsystem for operation in the batch or on-line environment. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, disk packs, and RAD storage.

PCL is controlled by single-line commands supplied through on-line terminal input or through command card input in the job stream. The command language provides for single or multiple file transfers with options for selecting, sequencing, formatting, and converting data records. Additional file maintenance and utility commands are provided. (References: CP-V/TS Reference Manual, 90 09 07, and CP-V/BP Reference Manual, 90 17 64.)

LEMUR

LEMUR (Library Editor and Maintenance Utility Routine) is a processor available in both on-line and batch modes. It builds and manipulates ROM and load module libraries. The libraries thus built are accessed by the LYNX or Load loaders when constructing user load modules. (CP-V/TS Reference Manual, 90 09 07, and CP-V/BP Reference Manual, 90 17 64.)

SYSGEN

SYSGEN is made up of several processors. These processors may generate a variety of CP-V systems that are tailored to the specific requirements of an installation. The SYSGEN processors are PASS2, LOCCT, PASS3, and DEF. PCL is used to select from various sources the relevant modules for system generation. PASS2 compiles the required dynamic tables for the resident monitor. LOCCT and PASS3 file away and execute load card images to produce load modules for the monitor and its processors. DEF writes a monitor system tape that may be booted and used. (Reference: CP-V/SM Reference Manual, 90 16 74.)

GENMD

GENMD permits on-line, batch, and ghost users to make permanent modifications to existing load modules, thereby reducing the number of compilations required to debug a program. (Reference: CP-V/SP Reference Manual, 90 31 13).

DEFCOM

DEFCOM makes the DEFs and their associated values in one load module available to another load module. It accomplishes this by using a load module as input and by producing another load module that contains only the DEFs and DEF values from the input module. The resultant load module of DEFs can then be combined with other load modules. DEFCOM is used extensively in constructing the monitor and the shared run-time libraries. (Reference: CP-V/BP Reference Manual, 90 16 64.)

SYMCON

The Symbol Control Processor (SYMCON) provides a means of controlling external symbols in a load module and of building a global symbol table. Its primary function is to give the programmer a means of preventing double definitions of external symbols. It may also be used to reduce the number of external symbols. For example, if certain load modules cannot be combined because their control tables are too large, the tables may be reduced in size by deleting all but essential external symbols. (Reference: CP-V/BP Reference Manual, 90 17 64.)

ANLZ

ANLZ provides the system programmer with a means of examining and analyzing the contents of dumps taken during system recovery. It is called automatically by the Automatic Recovery Procedure and is executed as a ghost job. It may also be called by the operator to analyze tape dumps when recovery is not possible, or by an on-line user to examine crash dumps or the currently running monitor. (Reference: CP-V/SP Reference Manual, 90 31 13.)

BATCH

The Batch processor is used to submit a file or a series of files to the batch queue for execution. Through Batch processor commands, the following capabilities are available:

1. A file may be inserted into a file being submitted for execution, thus bringing together more than one file to create a single job.

2. Selected strings and fields existing in files being submitted for execution may be replaced by new strings and fields.

3. The results of string and field replacements can be examined before the job is submitted to the batch stream.

4. Files to be submitted for execution may reside on tape or private disk pack.

5. Jobs may be submitted to run in an account other than the account from which the job is submitted.

The Batch processor may be called in either the on-line or the batch mode. (Reference: CP-V/TS Reference Manual, 90 09 07.)

DRSP

DRSP (Dynamic Replacement of Shared Processors) enables the system programmer to dynamically add, repace, or delete processors during normal system operation with other users in the system. (Reference: CP-V/SP Reference Manual, 90 31 13.)

ELLA

The Error Log Listing program (ELLA) provides an efficient tool to list and sort the error data base which is automatically generated and updated by the CP-V system. (Reference: CP-V/SP Reference Manual, 90 31 13.)

SHOW

The Show processor allows the user to display his current maximum system services and resources, the peripheral devices that he has been authorized to use, and several other system user parameters. (Reference: CP-V/SP Reference Manual, 90 31 13.)

The application processors are intended for use for specific types of applications.

## SORT/MERGE

The Sort/Merge processor provides the user with a fast, highly efficient method of sequencing a nonordered file. Sort may be called as a subroutine from within a user's program or as a batch processing job by control cards. It is designed to operate efficiently in a minimum hardware environment. Sorting can take place on from 1 to 16 keys and each individual key field may be sorted in ascending or descending sequence. The sorting technique used is that of replacement selection tournament and offers the user the flexibility of changing the blocking and logical record lengths in explicitly structured files to different values in the output file.

The principal highlights of Sort are as follows:

1. Sorting capability allows either magnetic tapes, disks, or both.
2. Linkages allow execution of user's own code.
3. Sorting on from 1 to 16 keys fields in ascending or descending sequence is allowed. Keys may be alphanumeric, binary, packed decimal, or zoned decimal data.
4. Records may be fixed or variable length.
5. Fixed length records may be blocked or unblocked.
6. Disks may be used as file input or output devices, or as intermediate storage devices.
7. Sort employs the read backward capability of the tape device to eliminate rewind time.
8. User-specified character collating sequence may be used.
9. Buffered input/output is used.

(Reference: Sort-Merge/Reference Manual, 90 11 99.)

## EDMS

EMDS is a generalized data management system that enables the user to create an integrated data base. It may be used with COBOL, FORTRAN, and Meta-Symbol processors. It simplifies programming by performing most of the I/O logic and data base management for the application programmer.

The principal features of EDMS are as follows:

- The user can describe data in various data structures. Using sets, any element can be related to any other element. The data structures include lists and hierarchies (trees). The two relationships can be combined to form extensive networks of data.
- Access techniques include random, direct, indexed, and indirect (relative to another record).
- An EDMS data base may consist of up to 64 monitor files.
- Multiple secondary indexes can be defined by the user to allow records to be retrieved via any combination of secondary record keys.
- Users may construct any number of logical files or data bases within an EDMS file.
- Data is described separately from the user program to facilitate management of the data base.
- Comprehensive security exists at all levels of a file.
- Journalization provides an audit trail for backup and recovery.
- A dynamic space inventory is maintained to facilitate rapid record storage and to optimize the use of available storage space.
- Detailed data description is provided for inclusion into the user's application program to reduce programming effort.

- File I/O logic is performed for the user program including

  1. Logical or physical record deletion.

  2. Record retrieval on random or search basis.

  3. Record insertion or modification.

(Reference: EDMS/Reference Manual, 90 30 12.)

## TRANSACTION PROCESSING

Transaction Processing is designed for applications that require the entry and processing of on-line transactions. It is a collection of general-purpose components and supporting monitor services available under the CP-V operating system. Transaction Processing (TP) enables a business to move from cyclic batch processing to remote on-line operations, where transactions are entered directly from their point of origin. The system consists of

- The CP-V monitor and standard processors such as COBOL, Meta-Symbol, and FORTRAN.

- Terminal Interface Controller.

- Utility processors that create files for external system control.

- Transaction Processing Controller.

- Extended Data Management System (EDMS).

(Reference: CP-V/TP Reference Manual, 90 31 12.)

## USER PROCESSORS

Users may write their own processors and add them to CP-V or replace CP-V processors. The rules governing the creation and modification of processors are described in CP-V/SP Reference Manual, 90 31 13.

# SYSTEM MANAGEMENT PROCESSORS

System management processors furnish the manager of a CP-V installation with on-line control of the system.

## SUPER

Super gives the system manager control over the entry of users and the privileges extended to users. Through the use of Super commands, the system manager may add and delete users, specify how much core and disk storage space a user will have, specify how many central site magnetic tape units a user will have, grant certain users, such as system programmers, special privileges, (e.g., the privilege of examining, accessing, and changing the monitor), and individually authorize or deny access to the various processors for each user. Super is also used to create and delete remote processing workstations. (Reference: CP-V/SM Reference Manual, 90 16 74.)

## CONTROL

The Control processor provides control over system performance. CP-V has a number of performance measurements built directly into the system. Commands of the Control processor enable the system manager to display these measurements and to "tune" the system as needed by setting new values for the parameters that control system performance. (Reference: CP-V/SM Reference Manual, 90 16 74.)

## RATES

The Rates processor allows the system manager to set relative charge weights on the utilization of system services. Specific items to which charge weights may be assigned include

1. CPU time.

2. CPU time multiplied by core size.

3. Terminal interactions.

4. I/O CALs.

5. Console minutes.

6. Tapes and packs mounted.

7. Page-date storage.

8. Peripheral I/O cards plus pages.

(Reference: CP-V/SM Reference Manual, 90 16 74.)

## FILL

The FILL processor performs three basic file maintenance functions:

1. It copies files from disk to tape as a backup.

2. It restores files from tape to disk.

3. It deletes files from disk.

(Reference: CP-V/OPS Reference Manual, 90 16 75.)

## FSAVE

The Fast Save (FSAVE) processor is designed to save disk files on tape at or near tape speed. The processor is faster than any other file saving procedure under CP-V. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

## FRES

The File Restore (FRES) processor is designed to restore to disk files that were saved on tape by FSAVE or Fill. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

## VOLINIT

VOLINIT provides for the initialization of public and private disk packs. It is used to establish serial numbers and ownership, to write headers and other system information in selected areas of the volumes, and to test the surface of the disks and select alternate tracks to be used in place of flawed tracks. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

LABEL

The Label processor initializes ANS tapes by writing ANS formatted labels. It may also be used to create "unlabeled" tapes from new tapes to be used as scratch tapes and to print the contents of the header and trailer labels of labeled tapes or the first 80 bytes of each block on unlabeled tapes. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

STATS

The STATS processor displays and collects performance data on a running system and produces snapshot files to be displayed by the report generator Summary. (Reference: CP-V/SM Reference Manual, 90 16 74.)

SUMMARY

The Summary processor provides a global view of system performance by formatting and displaying the statistical data collected by STATS. (Referene: CP-V/SM Reference Manual, 90 16 74.)

SYSCON

SYSCON is a system control processor that can be used to partition resources from the system, to return resources to the system, and to display the status of the various system resources. SYSCON can also be used to build, update, or display the M:MODNUM file, a file which contains device and controller model numbers. (Reference: CP-V/SM Reference Manual, 90 16 74.)

GRANULE ACCOUNTING CLEANUP PROCESSOR (GAC)

The Granule Accounting Cleanup (GAC) processor correlates information between the file DISKPOOL and the account authorization file, :USERS.. DISKPOOL is created by the FSAVE processor and contains specific account information. Each account record in DISKPOOL contains an entry for accumulated public disk pack granules and an entry for accumulated RAD granules. When GAC is run, these accumulated values are compared against the maximum values for the corresponding accounts in the :USERS file and the user's entry in the :USERS file is updated to reflect the latest accumulated values for RAD and disk. When the accumulated RAD or disk granules exceed the corresponding maximum values, this fact is noted in the report that is produced by the GAC processor. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

FIX

The Fix processor enables the system manager to repair or delete damaged file directories. It also provides HGP reconstruction for private disk pack sets and the public file system. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

DEVDMP

The Device Save/Restore processor (DEVDMP) is a stand-alone utility program designed to dump entire disk volumes to magnetic tapes for restoration at a later time. Restoration may only be made to an identical storage unit. (Reference: CP-V/OPS Reference Manual, 90 16 75.)

ONLIST

The ONLIST processor is invoked by a system management account in the batch, ghost, or on-line mode to display the contents of the :LOGD file used for the TEL WHERE command. This file, created and updated by LOGON, contains one record for each on-line user. The records are keyed by the users' sysid. Each record contains the user's line number, name, account, and the time the user logged on or off. Since LOGON accesses the :LOGD file in shared update mode, ONLIST should be used to list the file rather than PCL to avoid delaying LOGON. When invoked on-line, ONLIST displays only those users currently logged on. When invoked in batch or ghost mode, all records in the file (those of both logged on and logged off users) will be listed. In batch mode, records for logged off users are deleted from the file. These listings are produced simply by calling the ONLIST processor. No commands are required.

## PHYSICAL PAGE STEALER (PPS)

The Physical Page Stealer is a ghost job which is used for management of all dedicated foreground memory in real-time systems. PPS allows the user to display memory segments currently allocated, get DYNRESDF pages, free DYNRESDF pages, and redefine the RESDF area. (Reference: CP-V/SP Reference Manual, 90 31 13.)

## ACCOUNT X

This CP-V account contains a number of unsupported programs that were, for the most part, created in the process of developing CP-V. One of these programs (HELP) provides information about all of the other programs in account X. HELP is called by entering HELP.X in response to a TEL prompt (!). As soon as HELP is entered, it prints NEXT: on the terminal. If the user responds Y, HELP then types a brief description on the terminal, including a definition of the HELP commands.

Two of the HELP commands are List and Help. If the user enters the command L, HELP lists the names of all programs in account X. If the user enters the command H, HELP prompts with NAME= whereupon the user enters the name of the program for which he desires information. Other HELP commands provide additional information about account X programs.

## OTHER PROGRAMS

The Software Library Distribution Center that distributes CP-V contains a large number of useful programs that are not supported by the CP-V staff. These programs can be ordered from the library for execution under CP-V. This set of programs is listed in the Program Availability List (PAL manual).

Most of these programs are contributed by CP-V users and are supported by the users. Examples are SNOBOL and ALGOL.

# INDEX

Note: For each entry in this index, the number of the most significant page is listed first. Any pages thereafter are listed in numerical sequence.