

ESD-TR-71-74

AD718973

COMPUTER OPERATING SYSTEMS CAPABILITIES;
A SOURCE SELECTION AND ANALYSIS AID

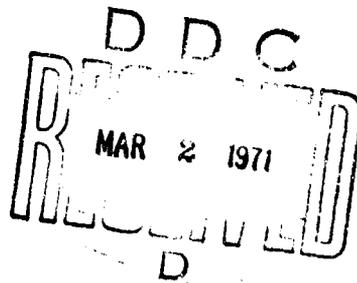
William C. Mittwede

November 1970



DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
HQ ELECTRONIC SYSTEMS DIVISION (AFSC)
L. G. Hanscom Field, Bedford, Massachusetts 01730

This document has been
approved for public release and
safe; its distribution is
unlimited.



(Prepared under Contract No. F19628-70-C-0258 by The COMTRE Corp.,
151 Sevilla Avenue, Coral Gables, Florida 33134.)

Reproduced by
**NATIONAL TECHNICAL
INFORMATION SERVICE**
Springfield, Va. 22151

PRICE SECTION

BILL SECTION

BY SPECIAL

BY STANDARD

LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

ESD-TR-71-74

COMPUTER OPERATING SYSTEMS CAPABILITIES;
A SOURCE SELECTION AND ANALYSIS AID

William C. Mittwede

November 1970

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
HQ ELECTRONIC SYSTEMS DIVISION (AFSC)
L. G. Hanscom Field, Bedford, Massachusetts 01730

This document has been
approved for public release and
sale; its distribution is
unlimited.

(Prepared under Contract No. F19628-70-C-0258 by The COMTRE Corp.,
151 Sevilla Avenue, Coral Gables, Florida 33134.)

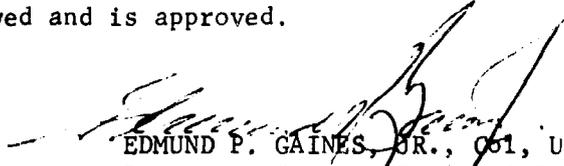


FOREWORD

This report presents the results of an analysis conducted by The COMTRE Corporation, 151 Sevilla Ave., Coral Gables, Florida. The analysis was conducted under Contract F19628-70-C-0258 with the Air Force Electronic Systems Division in support of Project 6917, Task 691701. Dr. John B. Goodenough, ESD/MCDS, was the ESD Contract Monitor.

This report has been reviewed and is approved.


ALFRED J. BEAUCHAMP
Project Officer


EDMUND P. GAINES, JR., O51, USAF
Director, Systems Design & Dev
Deputy for Command & Management Sys

ABSTRACT

This report presents a method for translating operational data processing requirements into specific criteria for use in selecting, validating or evaluating computer operating systems. The criteria have been structured on the basis of an integrated functional classification structure applicable to the executive/control functions, system management functions and data manipulation functions of currently available operating systems. In concert with the methodology presented, a checklist form is included as an aid to developing selection criteria for particular applications. A diagram of the functional classification structure is also included.

CONTENTS

| | | |
|--|--|-----|
| Section I | - Introduction | 1 |
| | 1.1 Purpose | 1 |
| | 1.2 Scope | 2 |
| Section II | - Specification and Evaluation Guidelines | 3 |
| | 2.1 Operating System Requirements Identification | 3 |
| | 2.2 Evaluation of the Environment | 3 |
| | 2.3 Basic Performance Decisions | 5 |
| | 2.4 Use of the Selection and Evaluation Criteria | 8 |
| Section III | - Specification and Evaluation Criteria | 11 |
| | 3.1 Introduction | 11 |
| | 3.2 Requirements Checklist - Part I - Executive/ Control Functions) | 11 |
| | 3.2.1 First Level of Detail (Part I - Executive/ Control Functions) | 11 |
| | 3.2.2 Second Level of Detail (Part I - Executive/ Control Functions) | 19 |
| | 3.2.3 Third Level of Detail (Part I - Executive/ Control Functions) | 43 |
| | 3.2.4 Fourth Level of Detail (Part I - Executive/ Control Functions) | 109 |
| | 3.3 Requirements Checklist - Part II - System Manage- ment Functions | 149 |
| | 3.3.1 First Level of Detail (Part II - System Management Functions) | 149 |
| | 3.3.2 Second Level of Detail (Part II - System Management Functions) | 159 |
| | 3.3.3 Third Level of Detail (Part II - System Management Functions) | 175 |
| | 3.4 Requirements Checklist - Part III - Data Manipulation Functions | 185 |
| | 3.4.1 First Level of Detail (Part III - Data Manipu- lation Functions) | 185 |
| | 3.4.2 Second Level of Detail (Part III - Data Man- ipulation Functions) | 193 |
| | 3.4.3 Third Level of Detail (Part III - Data Manipu- lation Functions) | 205 |
| | 3.4.4 Fourth Level of Detail (Part III - Data Man- ipulation Functions) | 235 |
| Attachment I - Computer Operating System Functional Classification Structure | | |

SECTION I

INTRODUCTION

1.1 Purpose

This report is the second of a series currently being produced by The COMTRE Corporation for the Electronic Systems Division of the Air Force Systems Command. The first report of this series, ESD-TR-70-377, presented an integrated functional classification structure applicable to the executive/control functions, system management functions, and data manipulation functions of currently available commercial operating systems. A third report will establish validation requirements for major computer operating systems.

In this report, criteria have been developed within the hierarchical structure of the functional classification presented in ESD-TR-70-377. Along with these criteria, methods for establishing a relationship between the criteria and operational requirements have been delineated for each criterion or group of associated criteria. These methods are intended to aid in specifying operating system requirements and preparing operating system specifications.

The analysis was based upon the following considerations relating to system specification practices:

1. Frequently criteria can be specified but are not; this is usually an oversight due to the lack of a comprehensive list of system specifications.
2. Each user tends to specify details for an area in direct proportion to his knowledge of that area; frequently, certain functions are slighted due to a lack of expertise in that area.
3. Features of aesthetic value, though not firm requirements, can often be specified to further enhance system usage.
4. The level of detail included within a specification can vary according to the intended purpose of the specification.

The method developed by this analysis provides a requirements checklist which will reduce the lack of specification due to oversights. Further, the requirements

checklist is keyed to references which will provide users information for areas in which they may not be experienced. And finally, the requirements checklist and references are structured into several levels of detail. While it is quite likely that the level of detail will vary for each section of a specification, the structure of the checklist is such that the required level of detail for each section can be reached in a methodical manner.

During the preparation of this report, the use of the requirements checklist and associated references as an aid in evaluating proposed systems became apparent. During the evaluation process the proposed operating system design can be compared against the system functional structure included within the checklist to ascertain completeness. Then, by using the reference material, the methods which appear to be most applicable to the given problem and to best satisfy the operational requirements can be systematically determined. The entire checklist and references can be used in this manner; however, levels three and four appear to be the most appropriate for detailed evaluation.

1.2 Scope

The analysis in this report relates operational performance requirements to specific operating system requirements for executive/control functions, system management functions and data manipulation functions. Requirements are related to functions in terms of specific criteria for performance specification or evaluation.

This report is presented in three sections with one supporting attachment. Section 2 presents a description of the techniques identified for determining operating system performance requirements. Section 3 presents the selection and evaluation criteria accompanied by explanatory references. Finally, a diagram depicting an overview of the operating system functional classification structure corresponding to the checklist levels of detail is attached to this report. This diagram can be used as a reference by personnel using the checklist to determine the association of the area in which they are working with the total classification structure.

SECTION II

SPECIFICATION AND EVALUATION GUIDELINES

2.1 Operating System Requirements Identification

The techniques for determining operating system performance requirements can best be described as a procedure consisting of three to six steps. The first step consists of evaluating the system environment. The second step consists of making basic system performance decisions. The third step consists of specifying performance criteria at a general level. The fourth, fifth and sixth steps amplify the third step at successively greater levels of detail.

The total number of steps used during the selection of any particular operating system varies depending upon the permissible level of detail for the particular selection. The appropriate level of detail, in turn, varies with the circumstances of a particular application of the selection technique. For example, if the technique is being applied for the purpose of developing an operating system specification for a known hardware configuration then the specification will probably be of a more detailed form than if the specification were prepared as part of a hardware solicitation.

2.2 Evaluation of the Environment

The first step in the procedure discussed above is an evaluation of the environment in which the operating system will be expected to perform. This step does no more than establish an overview of the intent of the system procedurally. It is, however, very important since it establishes the framework upon which most further decisions will be made.

Often, the essential capabilities necessary to satisfy a set of requirements can be envisioned conceptually or are known from past experience. In such cases an evaluation of this information will determine the basic characteristics of the operating system environment. In any case, however, the following environmental characteristics should be determined:

- o First, determine the salient characteristics of the processing modes which must be supported by the system.

It is very important in the further development of specific requirements to characterize system processing as real-time, batch, remote batch or time-sharing. It is also necessary to determine if a mixture of processing modes must be supported, e.g., time-sharing and batch, real time and batch, etc.

If the system is to perform computation that controls an ongoing process and delivers its outputs (or control inputs) not later than the time needed for effective control of the process, then the processing mode is real time. This mode is usually associated with air defense systems, ballistic missile checkout/control, space vehicle checkout/control, etc.

Most operating systems with the exception of some real-time processing systems allow the processing of jobs submitted to run independently of events outside the system on a deferred or time-independent basis (e.g., whenever the processing workload is light). If the system is to support this type of processing, then the processing mode is batch. Many times this feature is required for remote sites as well as the normal local site usage which means that the environment will also consist of a remote batch mode.

Finally, if the system is to support concurrent processing capabilities for several users via multiple terminals, then the mode is time-sharing.

- o Second, develop an application program scenario.

Although it has been determined by defining the processing mode that the application programs will be either batch/time-sharing/real time, there should also be many other known application support requirements. To formalize these requirements it is best to examine known applications or develop concepts relative to anticipated applications. Since the specification of requirements for an operating system depends to a large extent upon the satisfaction of application program requirements, it is necessary to develop an operational scenario for the application programs. This scenario should include such items as program descriptions, estimated program sizes, response time requirements, concurrency requirements, interaction requirements, interdependency requirements, operational hierarchy requirements, anticipated or known structural requirements, expected utilization, and unique features or requirements exhibited by the application programs.

- o Third, delineate all known hardware configuration information.

This consideration is simplified in areas where the hardware is known and a new operating system is being acquired. However, when a total hardware and software system is being acquired, as is often the case, this delineation can be quite difficult. Nevertheless, specification or postulation of a detailed hardware configuration should be accomplished if possible, since knowledge of a configuration is requisite to development of many specific criteria. The delineation of peripheral and communication devices is important as well as that of the processor configuration and the size or estimated size of the system. Within the framework of Section III of this report, reference is sometimes made to the size¹ of the system as a guide to applicable requirements. Previous analyses of various operating systems has shown that the occurrence of several features tends to be closely related to the system size.

The considerations listed above define the operating system environment, an application program scenario, and a specific hardware configuration. From this information, further decisions can be made regarding the expected performance of the system.

2.3 Basic Performance Decisions

As discussed above, there are certain decisions that should be made concerning the expected performance of the operating system. These decisions provide an insight into the total operational philosophy of the system and directly affect the applicability of specific requirements for specification or evaluation. The following topics and discussions are designed to aid in making these decisions.

- o Processing performance

Once the operating system environment has been established and the applications delineated, the next step should be to consider the need or lack of need for multi-

¹ For the purposes of this report, system size corresponds to the Operating System levels presented in ESD-TR-70-65, Survey and Analysis of Major Computing Operating Systems, 31 January 1970. These levels are:

- small scale computers with core storage generally less than 32K bytes (where a byte consists of 8 bits);
- medium scale computers with core storage ranging from 32K bytes to 132K bytes; and
- large scale computers with core storage in excess of 132K bytes.

program processing. If the application scenario previously developed indicates that the operating system must support concurrent core residence and processing of multiple programs, then multiprogramming is required. If there appears to be reason to expect that simultaneous execution of programs is required then this may dictate the consideration of a multiprocessor hardware configuration. However, if requirements indicate that concurrency of application operation is not an absolute prerequisite, then serial processing may also be acceptable.

The type of processing environment (batch, time-sharing, real time) can affect the operational philosophy of the system in the following respect. In a batch processing system, individual job throughput may not be of prime concern due to the usual background nature of the jobs supported; therefore, maximum utilization of system resources is usually more important than the amount of system overhead incurred. On the other hand, a time-sharing system would strive for a balance between resource utilization versus incurred overhead due to the requirement to interact and support multiple users. Finally, a real-time processing system usually stresses low overhead and requires maximum job throughput with resource utilization being of only secondary concern.

Therefore, tradeoff decisions need to be made relative to serial versus multiprogram processing, and overhead versus throughput versus resource utilization of the processing system.

o Mode of operational support

There are two basic modes of operation supported by operating systems. These are: continuous operational capability over an extended period of time and scheduled operation over a fixed period of time, e.g., eight-hour shift per day.

The major difference between these modes of operation is the criticality of the processing supported. Continuous operation is not normally a requirement for batch processing while it is almost mandatory for real-time processing and may be highly desirable for some time-sharing systems.

The determination of which of these modes of operation is to be provided by the system will aid in the determination of how comprehensive the error recovery requirements will be, whether on-line maintenance or periodically scheduled off-line maintenance will be required, and whether on-line or off-line program checkout will be

necessary. If the system is to provide the capability for continuous support with very little down time, then comprehensive error recovery procedures, on-line maintenance, and on-line program checkout should receive definite consideration. This decision can also affect the hardware configuration since equipment redundancy, either on-line or off-line, is frequently required for continuous support.

- o Type of user personnel

When it is known that the system must interface with and accommodate users with limited programming interests, such as in a general time-sharing system, greater attention should be paid to the area of diagnostics and system communication. This attention should focus upon the ease of usage, clear and simple messages, and clear and simple instructions.

If the system is real-time, the user can usually be assumed to be a sophisticated assembly or machine language programmer familiar with the internal organization of the computer; and although diagnostic and debug features are very important, they do not require the level of explanation and simplicity required for general time-sharing usage.

- o Type of checkout supported

If a system is to provide continuous operational support, then a method should be considered for allowing application program checkout concurrent with on-line operations. This would probably involve the utilization of a special checkout mode and therefore require special operating system support. This feature is required most frequently in real-time processing systems and is usually also standard in time-sharing processing systems.

- o Operational philosophy of the system

The question of manual intervention versus automatic decision making must be considered. Will the operating system be highly dependent upon operator intervention or will the operating system be as independent as possible requiring very little operator intervention?

- o Maintainability

Certain basic decisions should be made concerning the capabilities to maintain the system operationally, e.g., the ease of updating, changing, deleting, generating, and initializing, and the support to be provided for changing hardware configurations.

- o Reliability

This function is important to all systems but the real-time environment usually has the most stringent requirements in this area. Specifications for this area should consider the degree of editing, error checking, fault isolation, operational degradation, etc., that will be required by the operating system.

- o Expandability

Finally, it should be determined if the operating system may be required to perform additional functions in the future above and beyond those described within the application scenario. This decision can affect the selection of certain requirements which will ease the required operational transition at a later date.

When all of these considerations have been assessed, the next step is to select the requirements that will best satisfy the decisions made.

2.4 Use of the Selection and Evaluation Criteria

The third through sixth steps used in selecting operating system requirements are found within Section III of this report. Section III contains requirements checklists and references to aid in the selection of requirements for an operating system or for the evaluation of a proposed operating system. These requirements checklists are constructed within the framework established by the Operating System Functional Classification Structure. The entire structure as it relates to requirement specification is depicted in Attachment 1 to this report.

As shown in the attachment, the major operating system areas consist of: 1) Executive/Control Functions, 2) System Management Functions, and 3) Data Manipulation Functions. Each of these major functional areas is broken down into hierarchical levels of functions contained within the major functional areas. For example, within the major functional area Executive/Control the first level grouping is: 1.0 JOB MANAGEMENT, 2.0 DIAGNOSTIC ERROR PROCESSING, and 3.0 PROCESSING SUPPORT, while the second level grouping consists of such functions as 1.1 JOB CONTROL, 1.2 I/O CONTROL, 1.3 SYSTEM COMMUNICATION, 1.4 RECOVERY PROCESSING, 2.1 HARDWARE ERROR CONTROL, 2.2 PROGRAM ERROR CONTROL, etc. Each lower level is a more detailed functional breakdown of the functions of the preceding higher level. In certain functional areas this structure

is subdivided into only two levels, while in other areas it is subdivided into as many as four.

Similarly, the requirements checklist has also been structured by the same hierarchical levels and are presented for each of the three major functional areas (Executive/Control, System Management, and Data Manipulation).

The presentation of the functions in the group level format provides a procedural structure which allows personnel using the checklist to perform a step-by-step progression to the level of detail required for a particular application. Also, this type of structure provides a total system overview at each level. This is highly important in developing an understanding of the entire operating system structure. With this in mind, the attached diagram illustrating an overview of all levels of the hierarchical structure will also enable the user to relate each particular function to the composite structure.

The level to which the requirements are selected for specification is dependent upon many factors and an absolute rule can not be applied to determine the number of levels that should be used. In many cases the level to which the requirements can be specified is dependent upon the detailed knowledge or information available for particular system's applications. In other cases the level of specification may vary according to particular circumstances.

During the preparation of operating system specifications, it can be generally assumed that for a known hardware configuration the entire checklist should be used. In preparing a specification for an unknown hardware configuration, the first two levels are most appropriate, while caution should be exercised in specifying the requirements appearing in levels three and four. The reason for this is that many of the methods used to implement operating systems are directly related to the hardware upon which the operating system functions. Hence, many operating systems perform the same function using different methods. Consequently, delineation of specific requirements for operating system functions may lead to a choice between different implementation methods, the specification of either of which would be overly restrictive for a competitive procurement.

A word of caution should be interjected at this point: The requirements checklist is to be a working document used by personnel to indicate requirements for an operating system. A detailed review by the personnel preparing the final solicitation

document must be conducted to detect the specification of any improper or superficial requirements. The danger in using a checklist of the type proposed is that criteria can be specified, when in fact the criteria are not needed. Consequently, the user must continually recognize that valid justification is still necessary prior to the selection of any requirement.

Finally, it is a near certainty that although this is a fairly comprehensive checklist, there will still be some user requirements or methods of stating requirements other than those that appear within the checklist. These requirements, when they occur, should first be incorporated within the system specification by the user and along with available reference material should also become a permanent part of an updated checklist. This will insure that the checklist remains a comprehensive tool in performing operating system specification, selection and evaluation.

A user can use his own discretion in how he designates (e.g., yes, no, mandatory, optional, etc.) that a requirement is a criterion for his particular system. An example of this using page 44 (1.1.1 SCHEDULING) of this report as reference is as follows:

The user has an extremely complex scheduling requirement (several jobs competing for execution), so he would place a "yes" by requirement (a). Since he has several jobs in competition, he would place a "yes" by requirement (f), and if he knew the number of possible jobs in contention, he would fill in the blank within requirement (f), etc.

In many cases a requirement can be specified as "Mandatory" and this can be used as a designator by the user, or a requirement may be specified as "Optional" to mean that if a system has this feature, it is an added feature and will be a weighting factor during final system selection.

SECTION III
SPECIFICATION AND EVALUATION CRITERIA

3.1 Introduction

This section presents the operating system requirements checklist discussed in Section II. The checklist consists of a column of requirements applicable to each functional classification area, followed by three columns entitled "Reference", "Initial", and "Extended."

The "Reference" column contains reference numbers adjacent to the requirement or requirements to which they are applicable. These reference numbers coincide with the list of references on the facing page. These references will aid in the determination of whether the requirement should be specified or is applicable for specification for a particular operating system. As with the requirements checklist, the references are intended to be open-ended. The references should be extended as users identify additional considerations.

The column entitled "Initial" is to be used by personnel using the checklist to indicate whether a requirement is a criterion for the initial phase of the particular operating system which is being specified; while the column entitled "Extended" is to be used to indicate requirements that will be included within the operating system at a later date but are not required for initial installation. It should also be noted that within certain listed requirements blanks are available for inserting parameters when their values are known.

3.2 Requirements Checklist - Part I - Executive/Control Functions

The following subsections present specification and evaluation criteria for the components of the operating system that maintain real-time execution control over the system environment.

3.2.1 First Level of Detail (Part I - Executive/Control Functions)

This subsection covers the following first level executive/control functions:

- 1.0 JOB MANAGEMENT
- 2.0 DIAGNOSTIC ERROR PROCESSING
- 3.0 PROCESSING SUPPORT

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.0 JOB MANAGEMENT | 1 | | |
| (a) Batch processing support must be provided. | 2 | | |
| (b) Real-time support must be provided. | 2 | | |
| (c) Time-sharing support must be provided. | 2 | | |
| (d) Multiprogramming support must be provided. | 3 | | |
| (e) Multiprocessing support must be provided for _____ processors. | 4 | | |

1.0 JOB MANAGEMENT

Reference:

1. The job management function is responsible for the initiation, scheduling, monitoring, and control of system operations for all jobs submitted to the system. In this context, a job encompasses all of the programs required for the execution of a given application. The job management subfunctions consist of: job control, input/output control, system communication, and recovery processing.
2. The operational environment (batch, real-time, time-sharing) of a system is directly established by the intended system applications.
3. Multiprogramming is a technique that attempts to maximize computer throughput by interleaving the execution of two or more programs. Normally, multiprogramming is not a requirement as long as system throughput requirements can be met in a non-multiprogrammed manner. However, some systems require multiprogramming as a firm operational requirement without regard to throughput. These systems are normally those that combine two or more processing environments (such as batch and real-time processing) or those that are communication based systems using multiple terminals and requiring multiprogramming techniques due to the large number of concurrent messages received.
4. This criterion is entirely dependent upon the hardware configuration and can only be specified when the configuration is known.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 2.0 DIAGNOSTIC ERROR PROCESSING | 1 | | |
| (a) The system must provide hardware error control. | 2 | | |
| (b) The system must provide program error control. | 3 | | |
| (c) The system must provide interface error control. | 4 | | |
| (d) The system must provide error recovery procedures. | 5 | | |

2.0 DIAGNOSTIC ERROR PROCESSING

Reference:

1. The diagnostic error processing function is responsible for recognizing hardware, program and interface errors. Recognition is usually based upon hardware interrupts or program testable switches. The function also supports the diagnosis and resolution of error conditions.
2. This criterion is highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.
3. This criterion is rarely specified. However, it may be necessary to specify it when there will be a large amount of program testing in a batch processing or time-sharing environment or if the system operates in an on-line real-time environment.
4. Usually any interface that can exhibit control, introduce control, request control or initiate an executive function should be edited. The editing performed is generally based upon system defined format constraints.
5. Error recovery routines are usually provided with a system and they may be augmented by the installation during system generation. Augmentation or redesign of error recovery routines is most frequently found in real-time environments. This function should be specified for all processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 3.0 PROCESSING SUPPORT | 1 | | |
| (a) The system must provide a timing service. | 2 | | |
| (b) The system must provide testing/debugging service. | 3 | | |
| (c) The system must provide a logging and accounting service. | | | |
| (d) The system must provide system description maintenance. | 4 | | |

3.0 PROCESSING SUPPORT

Reference:

1. The processing support functions consist of supervisor routines which may be called upon to accomplish a variety of miscellaneous services for an application program. In general the services are utilitarian in nature and provide convenient, rather than necessary, functional support.
2. Most systems have an internal timing device which provides timing services to application programs. A few systems, to reduce the size of the resident supervisor, only include these services as an option during system generation. This feature is highly desirable in a real-time processing system and frequently occurs in both the batch processing and time-sharing environments.
3. In a real-time environment testing/debugging service specifications should take advantage of any specially provided hardware processing modes.
4. This criterion is rarely specified. However, it may be advisable when a single set of application and/or system programs are to be executed using two or more hardware configurations so that the programs may tailor themselves to the specific hardware or software environment.

3.2.2 Second Level of Detail (Part I - Executive/Control Functions)

This subsection covers the following second level executive/control functions:

- 1.1 JOB CONTROL
- 1.2 I/O CONTROL
- 1.3 SYSTEM COMMUNICATION
- 1.4 RECOVERY PROCESSING
- 2.1 HARDWARE ERROR CONTROL
- 2.2 PROGRAM ERROR CONTROL
- 2.3 INTERFACE ERROR CONTROL
- 3.1 TIMING SERVICE
- 3.2 TESTING/DEBUGGING SERVICE
- 3.3 LOGGING AND ACCOUNTING
- 3.4 PROGRAM ACCESSIBLE SYSTEM DESCRIPTION MAINTENANCE

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1 JOB CONTROL | 1 | | |
| (a) The system must provide support for the concurrent execution of up to ___ batch jobs. | 2 | | |
| (b-c) Batch support must be provided for: | | | |
| (b) central site job entry, | | | |
| (c) remote job entry. | | | |
| (d) The system must support ___ remote job entry batch terminals. | 3 | | |
| (e) The system must provide support for the concurrent execution of up to ___ real-time jobs. | 2 | | |
| (f) The maximum service time for a real-time request under average load conditions is _____. | 4 | | |
| (g) The maximum service time for a real-time request under maximum load conditions is _____. | 4 | | |
| (h) The system must provide control for real-time jobs initiated by clock interrupts. | | | |
| (i) The system must provide support for the concurrent execution of up to _____ time-sharing jobs. | 5 | | |
| (j) The system must support, at a maximum, simultaneous submission of jobs from _____ terminals. | | | |
| (k) The system must support, on the average, simultaneous submission of jobs from _____ terminals. | | | |
| (l-m) The system must provide a response to interactive terminals within a time frame | 4 | | |
| (l) of _____ during maximum load conditions, | | | |
| (m) of _____ during average load conditions. | | | |

1.1 JOB CONTROL

Reference:

1. Do not overlook any future expansion requirements in this area since it will probably be more economical to include these requirements in the basic system rather than to change the system at a later date.
2. While frequently employed for evaluation, this criterion is rarely specified. However, it may be necessary under the following circumstances:
 - a. enough information is known about the real-time environment to determine the number of independent real-time processes/interrupts requiring near-simultaneous servicing
 - b. sufficient information is known about the hardware configuration, throughput requirements, and projected batch processing load to dictate a level of accuracy.
3. This criterion is entirely dependent upon the hardware configuration and can only be specified when the configuration is known.
4. This criterion should be specified when system response time requirements are known.
5. This criterion should be specified for a time-sharing system.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | initial | Extended |
| 1.2 I/O CONTROL | 1,2 | | |
| (a) The system must support I/O scheduling. | 3 | | |
| (b) The system must provide support for up to ___ remote terminals. | 4 | | |
| (c) The system must permit the concurrent activity of up to ___ remote terminals. | 4,5 | | |
| (d) The system must provide support for up to ___ I/O devices. | | | |
| (e) The system must support the concurrent operation of up to ___ devices. | 6 | | |
| (f) The system must provide support for up to ___ I/O processors or channels. | 7 | | |
| (g) The system must support the concurrent operation of up to ___ I/O processors or channels. | 7 | | |
| (h-o) The system must support the following device types/ number of device types: | 7 | | |
| (h) ___ unit record devices, | | | |
| (i) ___ paper tape units, | | | |
| (j) ___ magnetic tape units, | | | |
| (k) ___ console typewriters, | | | |
| (l) ___ display devices, | | | |
| (m) ___ direct access storage devices, | | | |
| (n) ___ plotters, | | | |
| (o) ___ extended core storage. | | | |

1.2 I/O CONTROL

Reference:

1. System control over the activity of input and output devices is a characteristic feature of third-generation operating systems. This control is maintained for two separate reasons. First, it simplifies the work of the application programmer since he need not be concerned with the intricate details of programming for a variety of channel and device characteristics. This upgrades the overall effectiveness of the programming staff since a standard and well defined approach, rather than a number of widely varying approaches, is always used to interface with I/O devices. Secondly, since the system is in control of I/O activity, the application program need not be alerted to process I/O interrupts.
2. Future expansion requirements should be factored into the quantity of each device to be supported.
3. I/O scheduling is the process of acknowledging a request for I/O services and initiating the physical input or output operations to satisfy the request. Requests for service may be processed immediately or they may be serviced according to a queuing scheme. In the latter case, queues are provided to hold requests for channel or device services. This criterion should be specified for all system environments.
4. This criterion should be specified for either a time-sharing or remote batch processing system. It may only be specified in detail when the hardware configuration is known.
5. If it is possible that all remote terminals will be interacting or functioning with the system at any particular time, then the number of remote terminals will equal the number requiring concurrent activity.
6. This value should be determined by the anticipated utilization under peak loading conditions.
7. This criterion is highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3 SYSTEM COMMUNICATION | 1 | | |
| (a) The system must permit up to ___ operator terminals/ displays. | 2 | | |
| (b) The system must permit up to ___ user terminals/ displays. | 2 | | |
| (c-f) The system must support the following types of operator and user communication devices: | 2 | | |
| (c) card reader, | 3 | | |
| (d) console typewriter, | | | |
| (e) typewriter-CRT display, | 4 | | |
| (f) typewriter-printer. | 4 | | |
| (g) The system must provide device independent communi- cation formats. | 5 | | |
| (h) System startup must be provided. | | | |

1.3 SYSTEM COMMUNICATION

Reference:

1. System communication incorporates all of the functions involved in the exchange of information between the user or the computer operator and the operating system. The communication may be oriented either to controlling the execution of scheduled jobs within the system or to configuring system components and monitoring system status.
2. This criterion is dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known. Potential expansion requirements should also be considered.
3. The use of a card reader as a communication device is usually only associated with local or remote batch processing.
4. Typewriter printers and typewriter-CRT displays can be associated with both time-sharing and real-time environments.
5. Providing device independent communication formats entails much planning in the design phase; however, it is quite worthwhile to the user. This feature allows any communication device to be substituted for another device type without greatly affecting operation. Furthermore, this method of format standardization can be an economical factor in user job and program preparation. Consequently, it should be seriously considered during the development of specifications or during the evaluation of system proposals.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|------------------------------|--------------|----------|
| | | Initial | Extended |
| <p>1.4 RECOVERY PROCESSING</p> <p>(a-b) The system must provide checkpoint/restart facilities at the following levels:</p> <p>(a) program,</p> <p>(b) system.</p> <p>(c) The system must provide restart for all suspended programs.</p> | <p>1,2</p> <p>3</p> <p>3</p> | | |

1.4 RECOVERY PROCESSING

Reference:

1. Checkpoint/restart facilities are normally invoked whenever error processing routines have analyzed an error and determined that execution should be resumed from an earlier point in the processing cycle. Consequently, checkpoint/restart usually supplements normal error recovery operations and should be considered for specification in all system environments.
2. Checkpointing may be performed at either the system level or the program level. Only in rare circumstances are both provided. The system level checkpoints the entire system whereas the program level only checkpoints a single program. Consequently, in a real-time system checkpoint/restart facilities are generally initiated at the system level rather than the program level. In a batch processing or time-sharing system, on the other hand, checkpoint/restart facilities are most likely to be initiated at the program level.
3. When checkpointing is performed at the program level, a checkpoint may also be used to temporarily suspend and later resume an executing program in order to permit execution of one or more programs.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 2.1 HARDWARE ERROR CONTROL | 1 | | |
| (a) The system must provide for error correction. | 2 | | |
| (b) The system must provide error notification. | 3 | | |
| (c) The system must provide for recovery from hardware errors. | 4 | | |
| (d-i) The system must detect the following hardware errors: | 5 | | |
| (d) CPU errors, | | | |
| (e) I/O device errors, | | | |
| (f) I/O channel or I/O processor errors, | | | |
| (g) storage parity errors, | | | |
| (h) co-processor errors, | 6 | | |
| (i) power failure. | | | |

2.1 HARDWARE ERROR CONTROL

Reference:

1. It is usually necessary to specify this criterion when proposing an extensive error recovery scheme for a particularly complex system.
2. Generally, error correction is performed by retrying a failing operation and, if this fails, relying upon an alternate method of accomplishing the operation.
3. Usually, if error correction can be satisfactorily performed, notification of the error is not required. However, if the error can not be corrected, some form of error notification should be directed to the operator and to any affected interactive user.
4. System recovery from hardware errors can be associated with systems that support reconfiguration either through standby redundancy, replaceable modules and devices, or a degraded (fail soft) mode of operation. These functions are most frequently found in medium-to-large scale systems supporting a real-time environment where full time operation is required.
5. Indications of CPU errors, I/O device errors, I/O channel errors, parity errors, and power failures are generated by almost every hardware system.
6. The recognition of co-processor errors is only significant in a multiprocessor configuration. Generally, one of the non-failing processors will initiate diagnostics to determine the cause of the failure, the extent of the damage, and the recovery procedures that can be invoked. Error recovery for a multiprocessor configuration is more complicated (by an order of magnitude) than for uni-processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.2 PROGRAM ERROR CONTROL | | | |
| (a) The system must provide for program error correction. | 1 | | |
| (b) The system must provide program error notification. | 2 | | |
| (c) The system must provide controlled abnormal program terminations. | 3 | | |
| (d-j) The system must detect the following types of error: | 4 | | |
| (d) arithmetic errors, | | | |
| (e-g) instruction errors: | 5 | | |
| (e) invalid instruction, | | | |
| (f) unsupported instruction, | | | |
| (g) privileged instruction, | | | |
| (h) invalid address errors, | 6 | | |
| (i) storage protection errors, | 6 | | |
| (j) invalid data errors. | 7 | | |

2.2 PROGRAM ERROR CONTROL

Reference:

1. Almost all systems recognize program errors occurring in user programs and assume control to prevent the system from being adversely affected. The user is frequently allowed to supply his own error handling routines for certain types of errors, such as arithmetic and data errors.
2. Program error notification should be considered for a time-sharing system since the interactive user can frequently determine the corrective action that should be taken. In batch processing the error is normally logged and on-line notification is not normally performed. In the real-time environment when an error occurs in an operational program, it is usually desirable for notification to be made directly to the console operator.
3. This function is highly desirable in batch and time-sharing systems. In a real-time environment, a form of error recovery, rather than abnormal termination, should be considered.
4. These criteria are dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.
5. There are several different types of instruction errors that a system should recognize and distinguish. An invalid instruction is one that is not a part of the hardware's instruction repertoire. The normal error procedure should be to terminate the offending program. An unsupported instruction error occurs when a computer has optional instruction sets. Normally a programmed procedure can be used to simulate the optional instruction. Privileged instructions are used by most third generation systems to reserve a part of the instruction set for the sole use of the supervisory programs. By controlling the use of these instructions, application programs are much less likely to inadvertently damage the software system. The recognition or detection of an attempted use of one of these instructions by an application program should cause operator notification and possibly job termination.
6. Invalid addressing errors are detected by most systems when either the address does not physically exist within hardware storage or if it lies out of an application program's assigned working area. Unauthorized attempts to access OS resident areas or other user areas should be detected and the offending program should be terminated. In a shared computer system (batch or time-sharing), repeated occurrences should be brought to the attention of the operator. Time-sharing systems that process classified or private information should always storage protect this information and notify the operator of any violations.
7. It is usually impossible to determine invalid data content unless the user has specified limit values within which the data should occur. However, hardware detection can be used for detecting invalid data parity and adherence to device data formatting requirements.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 2.3 INTERFACE ERROR CONTROL | | | |
| (a) The system must edit operator key-ins. | 1 | | |
| (b) The system must edit input stream control commands. | 2 | | |
| (c) The system must edit remote terminal communications. | 3 | | |
| (d) The system must edit program to system linkages. | 4 | | |

2.3 INTERFACE ERROR CONTROL

Reference:

1. All processing system environments provide a form of operator communication. Most systems thoroughly edit and validate each operator input command prior to attempting to execute it since the failure to do so could result in a system failure. Consequently, serious consideration should be given to this criterion.
2. This criterion should be specified for a batch processing or time-sharing environment.
3. This criterion should be specified for time-sharing and remote batch processing environments.
4. This criterion is highly recommended for time-sharing and batch processing systems. It's usefulness in a real-time environment is somewhat questionable since real-time programs should be thoroughly validated prior to operational processing.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 3.1 TIMING SERVICE | 1 | | |
| (a) The system must provide real-time clock services. | 2 | | |
| (b) The system must provide interval timing services. | 3 | | |

3.1 TIMING SERVICE

Reference:

1. These features are highly desirable in a real-time processing system and are quite useful in time-sharing and batch processing environments. Implementation, however is dependent upon the availability of a timing device.
2. Real-time clock services are generally required for any environment in which the time of day will affect the processing workload. For example, in batch processing a time of day is frequently used as a deadline by which some processing jobs must be completed. In real-time systems, it may be used either for job scheduling or for distinguishing event occurrences (e.g., message time-stamping). Time-sharing systems have no particular requirement for a real-time clock.
3. Interval timers are generally required in a real-time environment in which execution scheduling is based upon a periodic interval (e.g., polling of communication lines). Interval timing services are also used by most time-sharing environments to control the execution time allotted to each user.

Interval timing service can also be incorporated within all systems as a debugging aid. One method of use is for an application program to set a time limit on the performance of a loop. If the time expires prior to loop completion or exit, then this indicates that something is wrong within the loop. Also, this function is sometimes used by systems to detect I/O devices that fail to respond within a certain designated time period.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 3.2 TESTING/DEBUGGING SERVICE | | | |
| (a) The system must provide storage dumps. | 1 | | |
| (b) The system must provide tracing facilities. | 2 | | |
| (c) The system must provide a special test/debug operating mode. | 3 | | |

3.2 TESTING/DEBUGGING SERVICE

Reference:

1. This criterion should be specified for all processing systems.
2. Tracing facilities are usually most extensive in a time-sharing system although they are also quite useful in testing batch and real-time programs. Specification of the criterion should be dependent upon the anticipated level of program testing that will be performed.
3. This criterion is highly desirable in both a batch and real-time processing environment. In real-time processing, it may permit some program testing while the system is actually "on-line," or it may allow the simulation of a real-time environment when the system is "off-line." It is also extremely useful in a batch processing system where a high degree of program testing occurs. In this area, it frequently takes the form of a set of pre-specified actions that can be invoked when a program error occurs. These actions override the system's standard abnormal termination processing capabilities.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|----------------------------|--------------|----------|
| | | Initial | Extended |
| <p>3.3 LOGGING AND ACCOUNTING</p> <p>(a-c) The system must maintain:</p> <ul style="list-style-type: none"> (a) job charge information, (b) error statistics, (c) system utilization statistics. | <p>1</p> <p>2</p> <p>3</p> | | |

3.3 LOGGING AND ACCOUNTING

References:

1. Batch processing and time-sharing operating systems normally require detailed accounting information on job execution time, device utilization, core utilization, etc. Consequently, this criterion should be specified for these two environments.
2. It is usually highly desirable to accumulate error statistics in order to identify hardware devices that have a greater than normal frequency of intermittent errors. As a result this criterion should be considered for all processing systems.
3. Though this criterion is rarely specified, it may be desirable to maintain system utilization statistics for relatively large and complex systems. These statistics in turn, can be examined to enable the system manager to tailor and tune various operating system functions to improve overall system performance.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 3.4 PROGRAM ACCESSIBLE SYSTEM DESCRIPTION MAINTENANCE | 1 | | |
| (a) The system must maintain current system status information. | 2 | | |
| (b) The system must maintain current system description information. | 3 | | |
| (c) The system must provide for the extraction of system description information by a user program. | 2 | | |

3.4 PROGRAM ACCESSIBLE SYSTEM DESCRIPTION MAINTENANCE

References:

1. Many batch processing and time-sharing systems maintain a certain amount of descriptive information in a supervisor communication region that may be interrogated by an application program. Three types of information are likely to be maintained: system defining information, current system status information and individual job information.
2. System status information may be of use to installation monitoring or accounting routines that are not built into the operating supervisor. Device status information (availability) is also extremely useful when an application program may use a number of different devices to accomplish its processing.
3. This information is useful when there are general purpose programs or compilers in operation that have the capability of alternate modes of operation based upon hardware and software status. For example, sort programs are usually designed to use all of the core area available.

3.2.3 Third Level of Detail (Part I - Executive/Control Functions)

This subsection covers the following third level executive/control functions:

- 1.1.1 SCHEDULING
- 1.1.2 RESOURCE ALLOCATION
- 1.1.3 PROGRAM LOADING
- 1.1.4 EVENT MONITORING
- 1.1.5 PROGRAM TERMINATION PROCESSING
- 1.2.1 I/O SCHEDULING
- 1.2.2 DATA TRANSFER
- 1.2.3 DEVICE MANIPULATION
- 1.2.4 REMOTE TERMINAL SUPPORT
- 1.3.1 SYSTEM STARTUP
- 1.3.2 JOB CONTROL COMMUNICATION
- 1.3.3 INPUT/OUTPUT STREAM CONTROL
- 1.3.4 RESOURCE STATUS MODIFICATION
- 1.3.5 SYSTEM STATUS INTERROGATION
- 1.4.1 CHECKPOINTING
- 1.4.2 RESTARTING
- 2.1.1 ERROR CORRECTION (Hardware errors)
- 2.1.2 ERROR NOTIFICATION (Hardware errors)
- 2.1.3 ERROR RECOVERY (Hardware errors)
- 2.2.1 ERROR CORRECTION (Program errors)
- 2.2.2 ERROR NOTIFICATION (Program errors)
- 2.2.3 PROGRAM TERMINATION
- 2.3.1 OPERATOR KEY-IN EDITING
- 2.3.2 CONTROL COMMAND EDITING
- 2.3.3 REMOTE TERMINAL COMMUNICATION EDITING
- 2.3.4 PROGRAM TO SYSTEM LINK VERIFICATION
- 3.1.1 REAL-TIME CLOCK SERVICE
- 3.1.2 INTERVAL TIMER SERVICE
- 3.2.1 STORAGE DUMP CONTROL
- 3.2.2 TRACING CONTROL
- 3.2.3 SYSTEM TEST MODE CONTROL
- 3.3.1 MAINTAINING JOB CHARGE INFORMATION
- 3.3.2 MAINTAINING ERROR STATISTICS
- 3.3.3 MAINTAINING SYSTEM UTILIZATION STATISTICS
- 3.4.1 CURRENT SYSTEM STATUS INTERROGATION
- 3.4.2 SYSTEM DEFINITION INTERROGATION

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.1 SCHEDULING | 1 | | |
| (a) The system must provide an algorithmic scheduling capability. | 2,7 | | |
| (b) The system must provide a time-initiated scheduling capability. | 3 | | |
| (c) The system must provide an event-initiated scheduling capability. | 4 | | |
| (d) The system must provide a program-initiated scheduling capability. | 5 | | |
| (e) The system must provide conditional scheduling. | 6 | | |
| (f) The system must recognize up to ___ scheduling priority levels. | 7 | | |
| (g-k) Batch scheduling must be permitted from the following sources: | | | |
| (g) local input stream, | | | |
| (h) remote input stream, | 8 | | |
| (i) an executing interactive job, | 9,5 | | |
| (j) an executing real-time job, | 9,5 | | |
| (k) another executing batch job. | 9,5 | | |
| (l) The system must provide scheduling for programs and/or subprograms which will be consistent with the desired sequence of operations. | 10 | | |

1.1.1 SCHEDULING

Reference:

1. The purpose of the scheduling function is to select a job which is available for processing and prepare it for execution. The function may be extremely complex, as in an extended multiprogramming system where several jobs may be simultaneously available for execution, or quite simple, as in serial processing systems where the order of programs in the input stream may dictate the execution sequence. Since most systems handle more than one type of processing mode, different scheduling philosophies are used for the real-time, time-sharing, and batch processing applications.

The greatest variation in the implementation of the scheduling facility exists in the handling of batch processing. The most elementary approach is to schedule each job for execution in the sequence of its occurrence in the input stream. When a system has separate input streams for several processing areas, as in serial processing or basic multiprogramming systems, each input stream serves as a scheduling queue which is external to the system.

Further sophistication of the sequential approach is achieved by pre-reading the entire input stream and storing it on a secondary storage device such as a disk. By so doing, jobs may be executed in an order other than input stream sequence. In this type of system, scheduling parameters are introduced to control the execution sequence. Normally, these parameters are priorities, where a higher priority job is executed before a lower priority job. Alternatively, the parameters may be clock times, where a job is initiated at a selected time or is initiated to be completed by a certain time. Clock-time scheduling may also be used to initiate selected real-time jobs.

The batch scheduling philosophy of an extended multiprogramming system allows jobs submitted from more than one input stream to compete for execution assignment. Under this philosophy a scheduling queue on an intermediate storage device is mandatory, and jobs are placed on this queue whenever they are entered from either a local or remote input terminal. In this type of system, an input stream symbiont is used to read the input stream and store it on the scheduling queue. The symbiont is itself scheduled by an external event such as an operator or user command to initiate input stream processing.

2. Algorithmic scheduling provides a priority scheduling concept where a number of factors may be allowed to influence the selection of the next program chosen for execution. This criterion should probably be specified for all extended multiprogramming systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.1 SCHEDULING (repeated) | 1 | | |
| (a) The system must provide an algorithmic scheduling capability. | 2,7 | | |
| (b) The system must provide a time-initiated scheduling capability. | 3 | | |
| (c) The system must provide an event-initiated scheduling capability. | 4 | | |
| (d) The system must provide a program-initiated scheduling capability. | 5 | | |
| (e) The system must provide conditional scheduling. | 6 | | |
| (f) The system must recognize up to ___ scheduling priority levels. | 7 | | |
| (g-k) Batch scheduling must be permitted from the following sources: | | | |
| (g) local input stream, | | | |
| (h) remote input stream, | 8 | | |
| (i) an executing interactive job, | 9,5 | | |
| (j) an executing real-time job, | 9,5 | | |
| (k) another executing batch job. | 9,5 | | |
| (l) The system must provide scheduling for programs and/or subprograms which will be consistent with the desired sequence of operations. | 10 | | |

1.1.1 SCHEDULING (cont'd.)

Reference:

3. Time initiated scheduling is frequently found in batch and real-time processing systems. It is used when job initiation must occur at a certain time or must be completed by a certain time. Clock time scheduling may also be used to initiate selected real-time jobs. Consequently, this criterion should be examined for a batch and real-time processing system. The criterion is rarely specified for a time-sharing system.
4. Initiating programs in response to events that produce an external signal to the compute. is the most straightforward of the scheduling methods. This criterion should normally be specified for both real-time and time-sharing systems.
5. Program-initiated scheduling permits an executing program task to request that another program task be scheduled for either asynchronous or subsequent execution. This capability frequently occurs in a time-sharing environment where background batch processing is initiated by a foreground time-sharing program. It is also found in large multiprogrammed batch processing systems.
6. Conditional scheduling permits the user to specify scheduling parameters which must be satisfied before the program can be initiated. These parameters tend to be the presence or absence of certain errors in a previous job step as well as the status of internally and externally set switches.
7. Many times the priority levels required to control scheduling can be determined from the environment. A batch environment will usually only have a few levels of priority to expedite urgent and/or short jobs. Time-sharing environments also generally need few priority levels. A real-time environment usually requires several levels of priority. These levels of priority are assigned to individual programs according to their execution requirements relative to other programs.

If the system is to operate in a mixed environment (e.g., batch and real-time, etc.) then a combination of priority levels for each environment will probably be required.
8. This criterion is dependent upon the hardware configuration and can only be specified when the hardware configuration is known.
9. A detailed examination of the intended or existing application programs must be performed to determine the desirability of this criterion.
10. This criterion should be specified when an operational scenario is included within the specification.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.2 RESOURCE ALLOCATION | 1 | | |
| (a-d) The system must allocate and prevent conflicts for, the following resources: | 2 | | |
| (a) core storage, | | | |
| (b) I/O devices, | | | |
| (c) data files, | | | |
| (d) common routines. | 3 | | |
| (e-g) The allocation of core storage must be by: | | | |
| (e) permanent assignment, | 4 | | |
| (f) static (job stream) requests, | 5 | | |
| (g) dynamic (execution time) requests. | 6 | | |
| (h-j) The allocation of I/O devices must be by: | | | |
| (h) permanent assignment, | 4 | | |
| (i) static (job stream) requests, | 5 | | |
| (j) dynamic (execution time) requests. | 6 | | |
| (k-l) The allocation of data files must be by: | | | |
| (k) static (job stream) requests, | 5 | | |
| (l) dynamic (execution time) requests. | 6 | | |

1.1.2 RESOURCE ALLOCATION

Reference:

1. The resource allocation function is responsible for assigning resources to each executing job in such a way that conflicting resource assignments are avoided. In general the system resources are CPU time, core storage, I/O devices, information files, and commonly used routines. The allocation of CPU time to a program is called dispatching and is covered separately under Section 1.1.4.
2. This feature is not necessary in a serial processing system since all system resources are normally made available to the executing program. However the criteria should be considered for both multiprogramming and time-sharing environments.
3. Routines that can be used by multiple programs may be designed to be loaded and executed when needed, or to be permanently core resident. They are rarely incorporated as a part of the executing program during the binding process except in serial processing or small multiprogramming systems.
4. This criterion is usually relevant for a basic multiprogramming environment where the user needs little control over resource assignment. It is primarily found in dedicated environments, such as real-time foreground/batch background applications in which the resources are permanently assigned to a particular environment.
5. This criterion is primarily relevant for those environments supporting input job streams (local and remote batch processing) and is also frequently found in time-sharing systems. The request for system resources occurring within the job stream generally means that the resources are assigned to the requesting element (job, job step, task) for the entire duration of the element's processing. With the exception of data files, these resources are generally not available for the use of other programs until this element terminates. Many systems will also not permit a program task to be scheduled until all static resource requests have been satisfied. The criterion should probably not be specified for real-time applications.
6. This feature occurs in many real-time environments as well as in large multiprogramming and time-sharing systems. The feature permits a system element to be assigned to an executing task only for the length it is actually needed, rather than for the duration of the entire task. A program will execute until it reaches a point at which a resource is required, request the resource, and then suspend itself until the resource becomes available. In a large system this reduces the number of I/O devices required and allows more effective utilization of core storage. It also invokes heavy overhead as well as introducing the possibility that a job in execution will be delayed because a resource is not available.

It is highly recommended for those systems where several programs may require access to a single on-line data base.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|----------------------------|--------------|----------|
| | | Initial | Extended |
| <p>1.1.3 PROGRAM LOADING</p> <p>(a) The system must provide for program or program segment loading into main memory.</p> <p>(b-e) The system must permit program loading from the following sources:</p> <p style="padding-left: 40px;">(b) system library, (c) user library, (d) input stream, (e) temporary library (e.g., compiler output file).</p> <p>(f) The system must provide facilities for absolute loading.</p> <p>(g) The system must provide facilities for relocatable loading.</p> | <p>1</p> <p>2</p> <p>3</p> | | |

1.1.3 PROGRAM LOADING

Reference:

1. The program loading function is responsible for loading a program task into core storage in such a way that it can be executed under the control of the system supervisor. There are two basic forms of program loading and either one or both is found in every operating system. The first, absolute loading, can only load a program that is in complete executable (core image) form. This technique requires very little system overhead during the loading process, though it also usually does require an independent support program element to convert the various programs into the executable core image format.

The second form of loading, relocatable loading, combines most of the functions of the independent support program element and the absolute loader into a single system element. A relocatable loader will assign preliminary storage addresses to the program, perform any address adjustments that may be required, combine the program with any required support subroutines, and produce a loaded executable program. The system overhead is considerably higher for this technique, however the human requirements for compiling, loading, and executing a program are greatly simplified.

As a result, absolute program loading is most generally found in small real-time control systems (where overhead must be minimized) and in small basic multi-programming and serial processing systems where the assigned program execution area is relatively static (such as a fixed background partition). Relocatable loaders occur most frequently in extended multiprogramming systems as well as in most time-sharing environments. Both loading techniques frequently co-exist in large multi-purpose systems.

2. Programs that may be loaded into core (using either technique) must reside in a defined location. Every system provides a system library which is composed of the operating system itself, the various program language compilers, and many of the most frequently used application programs. The system library is almost always on-line. Separate user libraries are generally designed to be removed when not in use. User libraries are also frequently used in large multi-user environments such as time-sharing and remote batch processing systems.

The loading of programs through the input stream is usually a supplement to libraries. This technique dates back to earlier computer systems where program decks were maintained apart from the computer and loaded only when needed. Most systems still retain the capability, but apart from remote batch processing applications it finds little use except in program testing.

3. The purpose of this feature is to protect the system and user libraries against the addition of unproven programs. Most systems provide this protection in some form. Therefore, the nature of the protection, and the specification of this criterion, would only be important in rare circumstances.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.4 EVENT MONITORING | 1 | | |
| (a) The system must provide fixed time-slice dispatching. | 2 | | |
| (b) The system must provide variable time-slice dispatching. | 2 | | |
| (c) The system must provide contention (priority) dispatching. | 2 | | |
| (d) The system must provide event synchronization. | 3 | | |
| (e) The system must recognize up to ___ distinct external interrupts. | 4 | | |
| (f) The system must recognize up to ___ interrupt levels. | 4 | | |
| (g) The system must provide limit monitoring. | 5 | | |

1.1.4 EVENT MONITORING

Reference:

1. Event monitoring refers to the control the operating system maintains over executing programs. The function includes: dispatching control, interrupt processing control, event synchronization, and program limit monitoring.
2. Dispatching is the supervisor controlled allocation of processor time to a specific task. Tasks that are eligible for dispatching have already been placed in an execution state by the scheduler and are not waiting for I/O activity, operator responses, etc. Dispatching is important only in multiprogramming and time-sharing.

Two techniques are frequently used: time-slicing and contention. Time-slicing allows one program to execute for a specified length of time, interrupts it, and selects another program to execute for another specified time period. Consequently, each program in execution is guaranteed a certain slice of time for execution. The contention technique, on the other hand, allows the highest priority program to execute until it no longer requires the CPU and then assigns the CPU to the next highest priority program. A low priority program is not guaranteed any execution time beyond that not used by higher priority programs.

Time-slicing is normally used for time-sharing whereas contention processing is almost mandatory for most real-time applications. Batch processing systems may use either technique, with little preference shown to one or the other.

While frequently used for evaluation, dispatching criteria are rarely specified. However, if an operating system is to be specifically designed for a given application, it may be desirable to consider the dispatching technique during initial criteria specification.

3. Event synchronization is the process of delaying task execution until some specified event occurs or the process of triggering a task upon the occurrence of a specified event. The most common types of events which may delay or initiate task execution are I/O completions, selected time intervals, subtask completions, and unsolicited key-ins.
4. This criterion is highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.

Interrupts that are provided in response to certain error conditions within the CPU (e.g., illegal instruction, arithmetic overflow, parity error, power failure, etc.) are considered internal interrupts and should not enter these calculations.

5. This criterion is frequently specified for time-sharing and batch processing in order to prevent misuse of system resources. The feature is generally desirable in a testing environment to assure that a program error does not result in voluminous output records, excessive CPU time, etc. Generally, limits are established for CPU time, output lines/cards/or records, and main and secondary storage space allocation.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-------------|--------------|----------|
| | | Initial | Extended |
| 1.1.5 PROGRAM TERMINATION PROCESSING | 1 | | |
| (a) The system must deallocate all resources at program termination. | 2 | | |
| (b-e) The system must provide summaries of the following information: (b) error statistics, (c) CPU time utilization, (d) device utilization, (e) file access statistics. | 3 | | |
| (f-i) The system must provide the following options at abnormal termination: (f) dump core, (g) dump files, (h) execute a specified program, (i) initiate recovery procedures. | 4 4 5 | | |
| (j) The system must notify the operator of abnormal terminations. | 6 | | |
| (k) The system must notify remote terminal users of abnormal terminations. | 7 | | |

1.1.5 PROGRAM TERMINATION PROCESSING

Reference:

1. A program terminates normally when it has completed all of its processing and returns control to the supervisor. A program may also be abnormally terminated by the operating system under a number of different circumstances. This is frequently caused by a programmed request for abnormal termination of the job, a system determination to abort due to lack of corrective actions for certain error conditions, or a console command to terminate issued by the computer operator. The standard abnormal termination procedure is to discontinue execution of the executing task, or possibly of the entire job, depending upon the criticality of the task with respect to the job.
2. This criterion should be specified for all processing systems that use the static resource allocation technique.
3. Summary status information is highly desirable for most batch and time-sharing applications. CPU time utilization, device utilization, and file access statistics can also be helpful to a facility in "tuning" the system to best meet operational requirements. Error statistics are an aid to hardware maintenance personnel.
4. This feature frequently occurs in a batch processing or time-sharing system as a means of providing program testing and debug support.
5. The initiation of recovery procedures is highly desirable and usually mandatory for most real-time processing systems.
6. This feature rarely occurs in time-sharing or large batch processing systems. When it does, it is normally restricted to operational programs rather than programs being tested. It should, however, be specified for a real-time processing system.
7. This criterion should be specified for time-sharing processing, remote batch processing and interactive real-time processing.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.1 I/O SCHEDULING | | | |
| (a) The system must provide immediate scheduling of I/O requests. | 1 | | |
| (b-e) The system must permit specific device assignment from the following system external sources: | | | |
| (b) input stream (control statement), | | | |
| (c) the operator, | 2 | | |
| (d) a program, | | | |
| (e) interactive user. | | | |
| (f) Specific device assignment must be the responsibility of the system. | 3 | | |
| (g) The system must provide queuing of input requests. | | | |
| (h) The system must permit specification of device priority. | 4 | | |
| (i) The system must permit the specification of I/O request priorities. | 5 | | |
| (j) The system must attempt to route I/O to a specific device via an alternate route if the primary route is busy or disabled. | 6 | | |
| (k) The system must provide facilities for alternate device selection. | 6 | | |
| (l-m) The system must provide facilities for initiating alternate device selection: | | | |
| (l) automatically, | | | |
| (m) by the operator. | 7 | | |

1.2.1 I/O SCHEDULING

Reference:

1. This criterion should be specified for all serial processing systems. This method is also used quite frequently in basic multiprogramming systems where dedicated real-time devices are known to be available. Any system supporting real-time processing where there are critical I/O operations should also consider this criterion.
2. Operator assignment of devices is highly desirable in most processing systems to permit an override of other assignment techniques due to unusual workloads.
3. Specific device assignment by the system is a dynamic method of selection which increases system throughput. This can be associated with multiprogram processing in which the system knows the status of all devices and can therefore make the best decision at a given moment as to which device should be made available to a particular program.
4. In certain real-time processing systems it is necessary to specify device priority due to the criticality of device operation, e.g. telemetry data, teleprocessing data, etc. Within some time-sharing systems, a requirement may also exist for some terminals to have priority over other terminals during system operation.
5. This criterion should be specified for most real-time processing systems.
6. This criterion is highly desirable for a real-time processing system; however, it is highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.
7. Automatic initiation is most useful within a time-critical environment or when there is a large number of alternate devices from which to make the selection.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | initial | Extended |
| 1.2.2 DATA TRANSFER | 1 | | |
| (a) The system must provide buffer control. | 2 | | |
| (b) The system must permit data code translation during data transfer. | 3 | | |
| (c-f) The system must process the following record types: | 4 | | |
| (c) fixed length records, | | | |
| (d) variable length records, | | | |
| (e) undefined length records, | | | |
| (f) character string records. | | | |
| (g) The system must accomplish all necessary code translation without the requirement for conversion or translation routines within applications programs. | | | |

1.2.2 DATA TRANSFER

Reference:

1. Data transfer controls the movement of input or output data between main storage and secondary storage, or between main storage and input/output devices. Prior to initiating the data transfer operation, an area of main storage (called a buffer) must be set aside. The buffer will either contain the output data to be transmitted or will receive the input data as it is being transmitted. Techniques for allocating buffer areas vary greatly among the various operating systems.
2. This criterion should be specified for all processing systems except certain small real-time processing systems in which the application program must perform all data transfer operations.
3. This requirement frequently occurs in teleprocessing where the input or output line data coding structures differ from the internal computer data codes. The requirement may also occur in real-time systems which are required to interface with analog devices. If the system will interface with any non-standard peripherals this criterion should also be considered.
4. An examination of the intended applications should determine if the OS is to support:

Fixed length records: Records having the same length as all other records in the same file.

Variable length records: Records having a length independent of the length of other records in the same file.

Undefined length records: Records having a length unspecified or unknown to the system.

Character string records: An unformatted record composed of a series of contiguous characters. Character string processing usually applies to teleprocessing messages.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| <p>1.2.3 DEVICE MANIPULATION</p> <p>(a-d) The system must provide facilities for:</p> <ul style="list-style-type: none"> (a) tape positioning, (b) disk positioning, (c) card stacking, (d) page ejecting. | 1 | | |

1.2. DEVICE MANIPULATION

Reference:

1. Device manipulation is a control function which allows a physical I/O device to be positioned without actually requiring data transfer. Device manipulation facilities normally permit volume positioning (rewinding, forward spacing, back-spacing, disk arm positioning, etc.), printer spacing and forms control, and card stacker selection. These features should be available on every operating system for the devices associated with the system.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.4 REMOTE TERMINAL SUPPORT | 1 | | |
| (a) The system must permit interactive communications. | | | |
| (b) The system must permit terminal to terminal communications. | 2 | | |
| (c) The system must permit terminal to operator communications. | 3 | | |
| (d) The system must permit operator control over remote terminal activity. | 4 | | |
| (e) The system must allow slaved remote terminals. | 5 | | |
| (f) The system must provide a remote batch communication mode. | 6 | | |

7.2.4 REMOTE TERMINAL SUPPORT

Reference:

1. Control over remote terminal operations is found in all time-sharing systems, interactive real-time systems and batch processing systems that provide remote job entry. While it may be possible to attach a remote terminal to practically any computer system, many operating systems are not designed to specifically support remote terminals and special terminal support routines must be designed to augment the normal supervisor facilities.
2. This feature is found in some time-sharing processing systems as well as in a few real-time environments. It is a very desirable feature for applications where remote users must interact with each other.
3. This feature should be specified for all processing systems that support remote terminals.
4. This feature is sometimes used to inhibit or lock-out certain terminals during processing of classified or private information. This feature may also be used to restrict the usage of certain terminals during critical (fail soft) operations.
5. Within certain system applications there exists the need to distribute or acquire data from terminals other than the prime terminal. If this is the case, then for proper operation certain terminals must be slaved to the prime terminal until completion of execution.
6. This criterion should be specified for all remote batch applications.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.1 SYSTEM STARTUP | 1 | | |
| (a) Startup of the entire system must be provided. | | | |
| (b) Startup on a partition by partition basis must be provided. | 2 | | |
| (c) The system must allow the use of cataloged startup procedures. | 3 | | |
| (d) The system must provide a capability during startup for respecification of parameters specified at system generation time. | 4 | | |
| (e) The system must permit specification of device availability during startup. | 5 | | |
| (f) The system must permit controlled system reconfiguration during startup. | 5 | | |
| (g) The system must provide full system restart procedures. | 6 | | |
| (h) The system must schedule user initiation programs. | 7 | | |
| (i) The system must request time/date specification. | | | |
| (j) The system must permit manual initiation of symbionts. | 8 | | |
| (k) The system must provide automatic initiation of symbionts. | 8 | | |

1.3.1 SYSTEM STARTUP

Reference:

1. System startup is performed by the computer operator to initialize the operating system for normal processing. In batch processing environments this is the normal beginning-of-the-day procedure once computer power has been turned on. In real-time environments operating around the clock, startup is only performed when the system has been shut down for some reason.
2. Startup on a partition by partition basis allows each partition to be started independently of the other partitions. When the system is divided into environment based partitions (batch, real time, time-sharing), then each area can be initiated without requiring the other areas. This feature is also associated with basic multiprogramming where each partition or group of partitions is supported by its own input stream.
3. This criterion should be specified where startup requirements are extensive and consistent.
4. This feature is highly desirable in most processing systems since it provides flexibility in tailoring a system to meet daily requirements.
5. This criterion should be specified for configurable processing systems.
6. System restarting is required when a failure that affects the total system, rather than a specific job, occurs. Restarting functions are oriented to restoring as much as possible of the system environment that was valid at the time of the error. In critical real-time environments, for example, system checkpoints are frequently taken at regular intervals. These checkpoints can be used to reload a previous valid version of the operating environment when no other immediate method of repair is possible.
7. This technique is highly desirable in a real-time processing system in which the system is required to interface with unique devices which can not be initialized using general initialization techniques. For example, user initiation programs may be required to selectively initiate teleprocessing operations.
8. Automatic symbiont initiation is generally advisable for output symbionts whereas manual initiation is more desirable for input symbionts.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.2 JOB CONTROL COMMUNICATION | 1 | | |
| (a-d) The system must provide control of user jobs from the following interactive sources: | 2 | | |
| (a) the batch job submitter, | | | |
| (b) the operator, | | | |
| (c) interactive users, | | | |
| (d) other executing jobs. | | | |
| (e) The system must permit up to _____ separate input stream devices. | 3 | | |
| (f) The system must provide for the use of cataloged procedures. | 4 | | |
| (g) The system must provide procedures for modifying cataloged procedures. | 4 | | |
| (h) The system must provide for conditional execution logic within the input stream. | 5 | | |
| (i) The system must provide the operator with the capability to redirect or abort output generated by a job initiated at a remote terminal. | | | |
| (j) The system must provide a job control language (JCL). | 6 | | |

1.3.2 JOB CONTROL COMMUNICATION

Reference:

1. Job control communication refers to that communication between the operating system supervisor and either the user or the computer operator relating to the initiation, running, or termination of individual jobs within the system. In batch processing systems, user/system communication is generally non-interactive, whereas in time-sharing systems it is almost always interactive. Operator/system communication is predominantly interactive.
2. Job submitter control occurs primarily in serial batch processing environments where the submitter has access to the operator console area.

In most batch environments the operator has the capability to exercise some control over user jobs, e.g. resource assignment, cancellation, etc. In a real-time environment, the operator should have the capability to exercise extensive control over executing jobs.

Most time-sharing and remote batch processing systems assign job control functions to interactive users.
3. This criterion is highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known. In a basic multiprogramming system this parameter is directly related to the number of partitions.
4. A cataloged procedure is a set of job control statements stored on a library and which may be invoked by being named on a special control card. This is an excellent feature where jobs are relatively standard or where the control language is complicated.

If cataloged procedures are used, then the flexibility should be available to allow modification of the procedures. This would be useful in diverting output from one device to another due to a new system configuration.
5. This feature is frequently found in larger batch processing systems. It allows non-interactive users to specify conditions for performing certain job steps. This feature is particularly useful in program testing and debugging.
6. This criterion should be specified for most batch processing and time-sharing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.3 INPUT/OUTPUT STREAM CONTROL | 1 | | |
| (a) The system must provide input stream control by symbiont processing. | 2 | | |
| (b) The system must provide output stream control by symbiont processing. | 2 | | |
| (c) The system must provide for the processing of up to ___ input streams. | 3 | | |
| (d) The system must allow up to ___ output streams to be maintained. | 3 | | |
| (e) The system must provide automatic editing of control command formats. | 4 | | |
| (f-j) The system must allow the following output stream elements: | | | |
| (f) diagnostic messages, | | | |
| (g) trace control listings, | | | |
| (h) data, | | | |
| (i) core dumps, | | | |
| (j) file dumps. | | | |

1.3.3 INPUT/OUTPUT STREAM CONTROL

Reference:

1. The input job stream is the sequence of batch processing control statements and program data submitted to the operating system on an input device specified for this purpose. In serial processing and basic multiprogramming systems, the device tends to be the system card reader, though, in fact, any input device can be used. In these two system types, jobs are read, processed, and output in the order in which they occur in the input stream.
2. In larger systems, particularly extended multiprogramming systems, the input and output streams are maintained as separate files in direct access storage. Programs called symbionts are used to read and transfer the system control and data cards from input devices to the input stream files. Other symbionts transfer output data from output stream files to the actual output devices.

The advantage of this technique can be best illustrated with an example. Consider the concurrent (either multiprogrammed or time-shared) execution of two independent application programs in a system that has a single system printer. If both programs require the use of the printer, only one can physically be assigned the device. If the device were assigned to both programs, output data from both jobs would appear intermixed in the listing. However, if one program is assigned the device, the other must be suspended until the device again becomes available. On the other hand, if both programs create separate output stream files on a direct access device, both programs can execute concurrently. When each program closes its output stream file, the file can be transferred to the printer by an output symbiont when the printer is available. Thus, the single system printer does not inhibit concurrent processing. A further benefit is that the system printer is not reserved during the entire execution period of either application program. Rather, it is reserved only for the length of time it takes to transfer the output data from secondary storage.

Thus, symbiont processing enables input/output devices to be utilized at physical data transfer rates, while permitting programs to process input and output stream data at storage file transfer speeds rather than at the lower peripheral device speeds. The overall effect on the system is a considerable increase in throughput without requiring additional peripheral devices.

Since, in time-sharing applications, the terminal is usually dedicated to a particular time-sharing job, and since both the input and output stream are usually located at the same terminal, no significant equipment or time saving is afforded time-sharing programs by using symbiont processing techniques. On the other hand, when the terminal is used for remote batch processing, symbiont processing can offer both time and equipment savings, particularly when multiple jobs are submitted.

3. This criterion is highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.
4. This criterion is highly desirable for all batch processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.4 RESOURCE STATUS MODIFICATION | 1 | | |
| (a-e) The system must provide control for on-line configuration modification of the following resources: | 2 | | |
| (a) peripheral devices, | | | |
| (b) mass storage allocation, | | | |
| (c) core storage allocation, | | | |
| (d) CPU time allocation, | | | |
| (e) input job queues. | | | |
| (f) The system must permit operator control of system configuration through operator console. | 1 | | |
| (g) The system must permit user control of system resource configuration. | 3 | | |
| (h-l) The system must recognize the following device conditions: | | | |
| (h) available, | | | |
| (i) down, | | | |
| (j) assigned, | | | |
| (k) reserved, | | | |
| (l) test mode. | 4 | | |
| (m-p) The system must permit the following types of resource modification: | 5 | | |
| (m) addition, | | | |
| (n) deletion, | | | |
| (o) replacement, | | | |
| (p) switching. | | | |
| (q) The system must allow reconfiguration in the event of malfunction and maintain continuity of operation. | 6 | | |

1.3.4 RESOURCE STATUS MODIFICATION

Reference:

1. In most computer operating environments, it is desirable to alter the computer configuration without physically terminating all system operations. For example, a tape drive may require cleaning, a disk file may require maintenance, or an off-line operation may have concluded and additional peripheral devices may have become available for system use. As a result, it is generally advisable to allow the computer operator to alter the status of resources available to the system during system operation. This is frequently accomplished via direct operator console commands to the operating system supervisor.
2. These criteria provide the flexibility to support changing workloads. The modification of peripheral device configuration is particularly advantageous where dynamic allocation or device substitution techniques are employed. Modification of mass storage allocation is desirable when this storage is used in support of real time or time-sharing.

On-line modification of core storage allocation is desirable in a system that supports fixed partitioning so that the allocation between foreground and background or real time/batch/time-sharing can be altered to satisfy changing requirements. This feature would not be relevant to serial processing, variable partitioned, or paged environments.

In multipurpose environments, it is sometimes desirable to alter the dispatching algorithm due to changing priorities or unusual system loads

In any environment in which the system supports input job queues, it is a necessity that some control be provided to modify these queues. This control should allow such functions as job deleting, postponing, replacing, etc. to be performed.

3. This feature is not prevalent due to the impact that user control could have on the total multiprogramming system operation. However, when the user does have dedicated resources (e.g., a remote batch terminal) then the criterion may be advisable.
4. Test mode recognition should be specified if on-line diagnostics are supported by the system.
5. Replacement is a manual technique while switching is normally used for automatic transfer to a standby on-line device.
6. This criterion is highly desirable for a real-time processing system.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.5 SYSTEM STATUS INTERROGATION | 1 | | |
| (a) The status of the system must be displayed continuously. | 2 | | |
| (b-c) The status of the system must be displayed upon request on a: | | | |
| (b) CRT, | | | |
| (c) printer. | | | |
| (d-i) The system must provide facilities to display the following information: | 3 | | |
| (d) identification of current users, | | | |
| (e) resource status, | | | |
| (f) job status (executing, waiting, suspended, etc.) | | | |
| (g) job information (priority, title, etc.) | | | |
| (h) input job queue status, | | | |
| (i) output job queue status. | | | |
| (j-m) The system must display causes of processing delays to include: | 4 | | |
| (j) peripheral non-availability, | | | |
| (k) data non-availability, | | | |
| (l) memory non-availability, | | | |
| (m) CPU delays due to unavailability of resources. | | | |

1.3.5 SYSTEM STATUS INTERROGATION

Reference:

1. The computer operator is usually given the capability of displaying the status of various system elements. This may range from the status of specific jobs to the status of I/O devices and main and secondary storage. Systems vary considerably in the capabilities provided and where some may only provide status information on particular items, others will produce extensive visual displays showing the current status, relative usage, and accumulated error statistics for any system element.
2. The display of system status continuously generally requires the use of a reserved CRT display device.
3. Many of these elements are valuable in monitoring the utilization of the system. It should be determined by the facility which will be most useful in their operating environment.
4. This feature is very important to a facility in determining where hardware or software changes can be applied to improve operation.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.4.1 CHECKPOINTING | 1 | | |
| (a-d) The system must provide checkpoint initiation from the following external sources: (a) control card, (b) system, (c) operator, (d) interactive user. | 2 | | |
| (e) The system must permit checkpoint initiation by the program itself. | 3 | | |
| (f) The system must permit checkpoint initiation by other programs. | 4 | | |
| (g-l) The system must provide checkpointing for the following program types: (g) process control programs, (h) teleprocessing programs, (i) interrupt handling programs, (j) interactive programs, (k) batch programs, (l) system supervisory programs. | 5 | | |
| (m-o) The system must allow checkpoint file maintenance on the following devices: (m) mass storage devices, (n) scratch tapes, (o) output data tapes. | 6 | | |
| (p) The system must allow multiple checkpoints to be maintained. | 7 | | |
| | 8 | | |

1.4.1 CHECKPOINTING

Reference:

1. Checkpointing is the recording of information about a program and its environment on secondary storage so that at any future time the program may be re-initiated from that point. Small operating systems, particularly serial processing systems, checkpoint all of core while most multiprogrammed systems checkpoint only individual storage partitions.
2. In batch processing systems, control card initiation of checkpoints sometimes occurs but is usually considered a poor substitute for programmed initiation. System initiated checkpointing is prevalent in small time-sharing systems where programs are continually being swapped between core and mass storage. System initiated checkpoints are highly desirable for real-time environments as an aid in recycling the system. However, while this is advantageous, it is not always possible and the intended real-time application should be examined to see if this is warranted. Operator initiated checkpoints are also provided for many processing systems.
3. This feature is highly desirable for all processing systems during program check-out and is sometimes mandatory in real-time processing systems to provide a recycle capability.
4. This feature frequently occurs in multiprogramming systems that support priority processing. A priority program requiring extra core can checkpoint another program and use its assigned core. When finished the priority program restarts the checkpointed program.
5. Checkpointing can be a very important feature in a real-time environment due to the possible requirement for continuous support. The checkpoint provides the system with the capability to perform quick recovery after malfunction.

This feature can be very advantageous in process control or interrupt handling but is extremely difficult to implement due to the unpredictable nature of the systems. In a system performing real-time monitoring, but not exercising control over external devices, implementation is considerably easier.

6. A checkpoint capability for interactive programs can be quite useful for program testing and debugging.
7. Checkpointing the supervisory programs is beneficial during early stages of operating system development if the area in which the supervisor resides is not storage protected.
8. The device upon which checkpoint files are maintained depends upon the hardware configuration. If the amount of mass storage available for checkpoint files is restricted, the use of scratch tapes should be considered. Mass storage usually only allows a few checkpoints to be maintained, while magnetic tapes allow multiple checkpoints. Checkpoint files may also be maintained on output data tapes. This has the advantage of automatically positioning the output tape when loading the checkpoint record for restart.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | | Reference | REQUIREMENTS | |
|--|---|-----------|--------------|----------|
| | | | Initial | Extended |
| 1.4.1 CHECKPOINTING (Cont'd.) | | | | |
| (q-u) | Checkpoint information recorded must include the following: <ul style="list-style-type: none"> (q) register contents, (r) I/O channel activity, (s) contents of core storage, (t) temporary data files, (u) permanent data files. | 9 | | |
| (v-y) | The system must allow checkpoint notification directed to the following recipients: <ul style="list-style-type: none"> (v) operator console, (w) interactive user terminal, (x) job output stream, (y) system log. | 10 | | |

1.4.1 CHECKPOINTING (cont'd.)

Reference:

9. It is necessary that all information required to restart a program be recorded. This information should usually consist of register contents, I/O channel activity, contents of core storage, and temporary data files. Permanent data files are usually not needed for checkpoint purposes unless they are being modified. Similarly, system mass storage files are not normally included within a checkpoint record. Temporary files in use by a program when the checkpoint occurs are normally required for proper program restart and should automatically be included as part of the checkpoint output.
10. If an interactive user has directed a checkpoint or initiated a checkpoint, then he should be notified.

When checkpoint notification is directed solely to the output stream it generally means that restart must be performed by a separate batch processing job.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.4.2 RESTARTING | 1 | | |
| (a-e) The system must permit restart initiation from the following sources: | | | |
| (a) job stream, | 2 | | |
| (b) program, | 3 | | |
| (c) operator console, | 4 | | |
| (d) system, | 4 | | |
| (e) interactive user terminal. | 5 | | |
| (f) The system must permit restart from the last checkpoint taken. | 6 | | |
| (g) The system must permit restart from any specified checkpoint. | 7 | | |
| (h) The system must permit checkpoints initiated on one system to be restarted on another system. | 8 | | |
| (i) The system must permit restart from the beginning of a job step. | 9 | | |
| (j) The system must provide automatic replacement of refreshable modules. | 10 | | |
| (k) The system must provide device repositioning. | 11 | | |

1.4.2 RESTARTING

Reference:

1. Restarting is the process of restoring the status of a job to some previous point in time. The three basic types of restart capabilities used within all operating systems are checkpoint, task, and job. A restart taken from a checkpoint re-establishes the program and its data in the operating environment as they existed when the checkpoint was taken and resumes execution at the restart address included in the checkpoint. A task restart re-initiates processing from the beginning of the identified task. A job restart re-initiates processing from the beginning of the entire job.
2. Restart initiation from the job stream is the normal procedure in a batch processing system where checkpoint notification is directed to the output stream.
3. Restart initiation by a program should be specified for those systems that support program initiated checkpointing.
4. System initiated restart is primarily a real-time processing function to attempt recovery after a malfunction or error.
5. When an interactive user has the capability to initiate checkpoints he should also have the capability to initiate restart.
6. This criterion should be specified for all processing systems supporting checkpoints.
7. If a multiple checkpoint capability is provided, this criterion should be specified.
8. This criterion is highly desirable when there is a separate computer system that may be used as a backup.
9. This technique frequently occurs in a batch processing system and is a fairly simple operation to perform.
10. A refreshable module is unique in that it is not modified by itself or any other program during execution. If there are indications that a program module has been damaged due to a hardware malfunction, then the system can replace a refreshable module with a duplicate copy without disturbing any intermediate data calculations.
11. Repositioning of all sequential devices at restart should be specified for all processing systems. It is not normally required for IAS devices.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 2.1.1 ERROR CORRECTION (hardware errors) | | | |
| (a-c) The number of retries of a failed operation must be: | | | |
| (a) fixed, | 1 | | |
| (b) determined at system generation time, | 2 | | |
| (c) determined by each user program. | 3 | | |
| (d-h) The system must provide control linkage to user error routines upon detection of the following errors: | 4 | | |
| (d) CPU errors, | | | |
| (e) I/O channel or I/O processor errors, | | | |
| (f) I/O device errors, | | | |
| (g) storage parity errors, | | | |
| (h) power failure. | | | |
| (i) The system must provide interactive correction procedures. | 5 | | |
| (j) The system must provide alternate I/O routing. | 6 | | |
| (k) The system must initiate an automatic diagnostic routine upon equipment malfunction. | 4 | | |

2.1.1 ERROR CORRECTION (hardware errors)

Reference:

1. Most systems provide some form of failed operation retry. The method most frequently used by all processing systems is a fixed number of retries.
2. This criterion is highly desirable for all processing systems since it provides a facility the flexibility to vary retry parameters based upon experience. This type of variation will usually occur due to different characteristics of magnetic tape or non-standard I/O devices.
3. This method is sometimes used in small to medium sized time-sharing systems and also real-time systems in which the user provides an I/O routine for unique device manipulation.
4. This function is highly desirable for a real-time processing system.
5. This feature rarely occurs in any processing system except as a means of permitting the operator to select from available options or to perform an override where redundancy is available. An example of override could be in the case of a real-time processing system which uses triple modular redundant voting circuits, one of which is continuously indicating error, where the operator instructs the system to disregard the error since two circuits are still functioning.
6. This criterion, while desirable for a real-time processing system, is highly dependent upon the hardware configuration.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.1.2 ERROR NOTIFICATION (hardware errors) | | | |
| (a) The system must output operator console error messages. | 1 | | |
| (b) The system must output interactive user console error messages. | 1 | | |
| (c) The system must permit subroutines and tasks to return error codes to calling programs. | 2 | | |
| (d) The system must update and maintain error statistics files. | 3 | | |
| (e) The system must provide diagnostic logout of permanent errors. | 3,4 | | |
| (f) The system must provide an error trace showing the events leading up to an error. | 3 | | |
| (g) The system must notify remote users of equipment malfunctions. | | | |
| 2.1.3 ERROR RECOVERY (hardware errors) | | | |
| (a-b) The system must provide the following forms of system reconfiguration: (a) alternate device utilization, (b) controlled system degradation. | 1 | | |
| (c-d) The system must allow manual reconfiguration through: (c) resource respecification, (d) system regeneration. | 1 | | |
| (e) The system must permit on-line system maintenance of devices. | 2,3 | | |
| (f) The system must provide for automatic restart from a system maintained checkpoint. | 2 | | |

2.1.2 ERROR NOTIFICATION (hardware errors)

Reference:

1. This criterion is highly desirable for all processing systems. Notification should usually occur if an error cannot be corrected or if it is recurring persistently.
2. This feature is desirable for most processing systems since continued operation is usually conditional upon error notification. Other alternatives to this feature would be automatic task abort or, if continued operation is required, operator/user notification.
3. This function is highly desirable for all processing systems but is usually found only in medium to large scale systems. This is a definite aid for hardware maintenance and system trouble shooting.
4. In a multiprocessor configuration, a separate processor can look at a logout from another processor to attempt to effect recovery.

2.1.3 ERROR RECOVERY (hardware errors)

Reference:

1. This criterion is highly dependent upon the hardware configuration. The alternate device mode is considered the better of the two modes since it does not affect the capability of the system to provide total support. The degraded mode allows the system to support the most critical operations while other operations are suspended. In critical real-time control systems the alternate device mode is used as a first resort and the degraded mode is used only when no alternate devices are available.

Either of these modes can be either manual or automatic, however, the automatic mode is used most frequently. Resource specification is the best method for performing manual reconfiguration while system regeneration is generally slow and used only as a last resort.
2. This feature is highly desirable for a real-time processing system.
3. This function can help reduce system down-time for scheduled maintenance. However, it usually requires a system that supports device pools or dynamic allocation of devices.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.2.1 ERROR CORRECTION (program errors) | | | |
| (a-g) The system must provide controlled linkage to user error routines upon detection of the following errors: | 1 | | |
| (a) arithmetic errors, | | | |
| (b-d) instruction errors: | | | |
| (b) invalid instruction, | 2 | | |
| (c) unsupported instruction, | | | |
| (d) privileged instruction. | | | |
| (e) invalid address errors, | | | |
| (f) storage protection errors, | | | |
| (g) invalid data errors. | | | |
| (h) The system must permit the inclusion of user defined instruction modules for unsupported instructions. | 2 | | |
| (i) The system must provide interactive correction procedures. | 3 | | |
| 2.2.2 ERROR NOTIFICATION (program errors) | | | |
| (a) The system must output program error messages on the operator's console. | 1 | | |
| (b) The system must provide abnormal termination indicators. | 2 | | |
| (c) The system must permit job steps to set error indicators for subsequent job steps. | 3 | | |
| (d) The system must provide program error notification to interactive users. | 4 | | |

2.2.1 ERROR CORRECTION (program errors)

References:

1. Most frequently linkage is provided to user error routines due to arithmetic or invalid data errors, while other errors tend to be handled by the system itself.
2. This criterion also provides the user with the capability to define and develop his own unique instructions.
3. This function frequently occurs in small real-time and time-sharing processing systems in which the programmer/user participates in an on-line mode during program debug cycles.

2.2.2 ERROR NOTIFICATION (program errors)

Reference:

1. This criterion, while highly desirable for a real-time processing system, is not usually desirable in a time-sharing or multiprogrammed batch processing system.
2. This function is highly desirable for all processing systems.
3. This feature, while highly desirable for a batch processing system, does not afford any advantage in a real-time or time-sharing application.
4. This criterion is highly desirable for time-sharing processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.2.3 PROGRAM TERMINATION | | | |
| (a) The system must provide conditional termination based upon a specified error level being reached. | 1 | | |
| (b-e) The system must allow abnormal termination to be initiated by: | | | |
| (b) the system, | 2 | | |
| (c) the operator, | 2 | | |
| (d) a user program, | | | |
| (e) an interactive user. | 3 | | |
| 2.3.1 OPERATOR KEY-IN EDITING | | | |
| (a) The system must provide assumed default options. | 1 | | |
| (b-d) The system must provide the following forms of error notification: | 2 | | |
| (b) coded messages, | | | |
| (c) free format messages, | | | |
| (d) tutorial messages. | | | |

2.2.3 PROGRAM TERMINATION

Reference:

1. This feature is highly desirable for all processing systems as an aid during initial program compilation and checkout.
2. Abnormal termination by the system is frequently used when system rules are violated e.g., privileged instruction, memory protect, or in a real-time processing system due to the invocation of a degraded mode of operation.

If the computer operator receives program error notification, he should also have the capability to initiate abnormal termination.

3. This criterion should be specified for a time-sharing processing system.

2.3.1 OPERATOR KEY-IN EDITING

Reference:

1. This criterion should be considered for specification for all processing systems since it will expedite operator key-ins.
2. While frequently employed for evaluation, these criteria are rarely specified. Generally, if the hardware configuration is small, then coded messages are used since they do not require much storage area. In larger systems, free format or tutorial messages may also be provided. When available, the tutorial presentation is the better of the two.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-------------------------------------|--------------|----------|
| | | Initial | Extended |
| <p>2.3.2 CONTROL COMMAND EDITING</p> <p>(a-c) The system must provide the following options upon detection of a control command error:</p> <ul style="list-style-type: none"> (a) job termination, (b) command rejection, (c) request for clarification by the operator/user. | <p>1</p> <p>2</p> <p>3</p> <p>3</p> | | |
| <p>2.3.3 REMOTE TERMINAL COMMUNICATION EDITING</p> <p>(a-d) The system must provide editing at the following levels:</p> <ul style="list-style-type: none"> (a) message format, (b) command structure, (c) data structure, (d) invalid characters. <p>(e-g) The system must provide the following types of error notification:</p> <ul style="list-style-type: none"> (e) coded messages, (f) free format messages, (g) tutorial messages. <p>(h) The system must edit the user/terminal ID.</p> | <p>1</p> <p>2</p> | | |
| <p>2.3.4 PROGRAM TO SYSTEM LINK VERIFICATION</p> <p>(a-c) The system must provide linkage verification at the following levels:</p> <ul style="list-style-type: none"> (a) requested operation code, (b) parameter list validation, (c) calling sequence. | <p>1</p> | | |

2.3.2 CONTROL COMMAND EDITING

Reference:

1. A mixture of these criterion may be provided based upon the command type.
2. This function frequently occurs in batch processing systems and is desirable for all non-interactive processing.
3. This criterion is highly desirable for interactive processing applications.

2.3.3 REMOTE TERMINAL COMMUNICATION EDITING

Reference:

1. These criteria are highly desirable for a time-sharing or remote batch processing system.

The edit levels most frequently supported (in the order of frequency at which support occurs) are: data structure, message format, command structure, and invalid characters.
2. Error notification should be specified for remote batch and time-sharing processing systems, with coded messages being most prevalent in small systems, free format messages prevalent in large remote batch systems, and tutorial messages in large time-sharing systems.

2.3.4 PROGRAM TO SYSTEM LINK VERIFICATION

Reference:

1. These criteria should be considered for most processing systems since the degree to which linkage verification is performed can affect system reliability.

When employed in evaluation, the system providing the most levels of verification should be favored.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 3.1.1 REAL-TIME CLOCK SERVICE | | | |
| (a) The system must provide the current date. | 1 | | |
| (b-d) The system must provide the time of day in the following formats: | 1 | | |
| (b) hours/minutes/seconds, | | | |
| (c) hours and decimal fraction, | | | |
| (d) internal code. | | | |
| (e) The system must provide date conversion facilities. | 2 | | |
| (f) The system must provide time format conversion facilities. | 2 | | |
| (g) The system must provide facilities for task interruption at a specified time. | 3 | | |
| (h) The system must provide facilities for task suspension until a specified time. | 3 | | |

3.1.1 REAL-TIME CLOCK SERVICE

Reference:

1. This criterion should be specified for any system that either uses job accounting facilities or time-initiated scheduling.
2. This criterion is highly desirable since it is usually more advantageous for the system to perform conversion than each user program.
3. This feature usually occurs in multiprogramming systems. An interrupt at a specified time is useful to a program that is to acquire data at an exact time of day e.g., acquisition of deep space telemetry data, etc. Suspension of a program until a specified time is another way of acquiring data that arrives at a specified time. In this way the program performs all execution up to a point at which the data is required and then is suspended.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 3.1.2 INTERVAL TIMER SERVICE | | | |
| (a) The system must provide up to ___ interval timers (fixed or programmable). | 1 | | |
| (b-c) The system must provide interrupt service as follows: | | | |
| (b) at the completion of a specified single interval. | 2 | | |
| (c) at each completion of a specified periodic interval. | 2,3 | | |
| (d-f) The system must permit time intervals to be measured in terms of: | | | |
| (d) actual elapsed time, | 2,3 | | |
| (e) elapsed task execution time, | 3,4 | | |
| (f) time the task is actually using the CPU. | 3,4 | | |
| (g) The system must permit task suspension for a specified time interval. | 2 | | |
| (h-m) The timer base (update interval) must be: | 5 | | |
| (h) the same for all interval timers, | 6 | | |
| (i) independent for each timer, | 6 | | |
| (j) fixed, | 6 | | |
| (k) specified at system generation time, | 7 | | |
| (l) specified by control commands, | 8 | | |
| (m) specified dynamically. | 9 | | |

3.1.2 INTERVAL TIMER SERVICE

Reference:

1. The number of interval timers a system must support is a function of how the interval timers will be used and the anticipated frequency of utilization. Since the interval timer is a hardware item, the number supported by the system is frequently established by hardware.
2. This criterion should be specified for a real-time processing system.
3. This function frequently occurs in a time-sharing processing system.
4. This feature frequently occurs in systems that perform an accounting function.
5. The precision of an update interval can usually be expressed in nanoseconds, microseconds, milliseconds, or seconds. A real-time processing system will usually be satisfied by microsecond or millisecond updates while a time-sharing processing system will usually be satisfied by millisecond or second updates.
6. Having the same update interval or a fixed timer base for all interval timers is the simplest hardware implementation method but it may not be flexible enough to support a combined time sharing and real-time processing system. An independent update interval for each interval timer provides quite a bit of flexibility as long as the intervals available cover the realm of application possibilities.
7. This criterion provides a great deal of flexibility if each timer update interval can be specified independently.
8. This criterion provides even greater flexibility on a day to day basis.
9. This method is the most flexible means available and should be considered when the interval timers are considered allocatable resources within a multi-programmed real-time processing system.

3.2.1 STORAGE DUMP CONTROL

Reference:

1. The features incorporated under storage dump control should be carefully evaluated in respect to the type and amount of program testing that is anticipated. Since the checkout cycle can be significantly improved by the implementation of many of these techniques, an installation anticipating continued program development would do well to assess this area in detail. On the other hand, an installation with little involvement in program development would only require a minimum of these facilities.
2. This criterion, if specified, should be restricted to use by facility OS maintenance personnel.
3. This criterion should be considered for a real-time processing system.
4. This criterion is highly desirable for both batch and time-sharing processing system.
5. This feature is highly desirable for all processing systems as an aid in the checkout cycle since the programmer/user will not be required to interpret or convert manually.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | | REQUIREMENTS | | |
|--|--|--------------|---------|----------|
| | | Reference | Initial | Extended |
| 3.2.1 STORAGE DUMP CONTROL (cont'd.) | | | | |
| (t) | The system must provide conditional dump display facilities, i.e., dump to high speed device for optional display. | 6 | | |
| (u) | The system must provide automatic dump initiation. | 7 | | |
| (v) | The system must provide conditional dump initiation. | 8 | | |
| (w-y) | The system must allow dump initiation from: | | | |
| (w) | control statements, | | | |
| (x) | operator key-in, | | | |
| (y) | interactive user key-in. | | | |

3.2.1 STORAGE DUMP CONTROL (cont'd.)

Reference:

6. This criterion should be considered for specification if a multiple dump capability is provided since it will give the user selectivity in displaying only the dumps or portions of dumps that he desires.
7. This feature frequently occurs in all processing systems at abnormal termination.
8. This feature frequently occurs in processing systems that provide error code levels, etc.

3.2.2 TRACING CONTROL

Reference:

1. These criteria are frequently employed in the evaluation of a tracing package and can only be specified if detailed information is available on the types of traces that will be required. Data tracing traces only references to specified data locations, instruction tracing traces every instruction, logic tracing traces only changes in instruction sequence (e.g., branches), supervisor service request tracing traces only system service request occurrences, and subroutine call tracing traces only entry and exit from subroutines.
2. This criterion should be considered for a batch processing system.
3. This criterion should be considered for a real-time processing system.
4. This criterion should be considered for a time-sharing processing system.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 3.2.3 SYSTEM TEST MODE CONTROL | | | |
| (a-d) The system must provide I/O simulation facilities to: | 1 | | |
| (a) ignore I/O requests, | | | |
| (b) reroute I/O requests, | | | |
| (c) log I/O requests, | | | |
| (d) simulate I/O error conditions. | | | |
| (e) The system must provide automatic storage dumps. | 2 | | |
| (f) The system must provide automatic file dumps. | 2 | | |
| (g) The system must allow the user to override abnormal abort services. | 3 | | |
| (h) The system must allow the user to override subsequent job step cancellation. | 3 | | |
| (i) The system must allow for the insertion of breakpoints in programs. | 4 | | |
| (j) The system must allow the user to start or restart a program at a specified address. | 4 | | |
| (k) The system must permit memory searching/display. | 4 | | |
| (l) The system must permit memory modification. | 4 | | |
| (m) The system must provide interrupt or error notification and allow the user to override those conditions. | 4 | | |
| (n-p) The system must provide for on-line program development from remote terminals to include: | 4 | | |
| (n) statement by statement entry of program or job, | | | |
| (o) compilation or assembly with return diagnostics, | | | |
| (p) line update or modification of a permanent or temporary program. | | | |

3.2.3 SYSTEM TEST MODE CONTROL

Reference:

1. When a system has a test mode that permits program or hardware testing while on-line support is also being provided, it is usually necessary for I/O simulation facilities to be provided to keep test programs from interfering with actual on-line processing. Simulation of I/O can also be a very important feature in performing program checkout prior to total system installation.

The different types of simulation techniques are rarely specified; however, they should be used as a means of evaluating a system's capability. The best method employed is rerouting of I/O requests, second is logging I/O requests, and last is ignoring I/O requests. The simulation of I/O error conditions is also important in that it allows diagnostic and error processing routines to be checked without actually introducing the hardware error.

2. This criterion should be specified for all processing systems as a means of supporting abnormal termination.
3. This criterion should be specified for all processing systems that provide abnormal termination services, since the services may not be required or wanted by the user each time abnormal termination occurs.
4. These criteria are highly desirable in all processing systems in which interactive testing is performed.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|---|--------------|----------|
| | | Initial | Extended |
| <p>3.3.1 MAINTAINING JOB CHARGE INFORMATION</p> <p>(a-g) The system must record and maintain the following job charge information:</p> <ul style="list-style-type: none"> (a) CPU time utilization, (b) I/O channel and device time utilization, (c) I/O record counts, (d) main storage utilization, (e) secondary storage utilization, (f) remote terminal utilization, (g) job termination conditions, (h) total elapsed time, (i) date. <p>(j) The system must provide linkage to user supplied accounting routines.</p> <p>(k-n) The system must allow job charge information to be placed on:</p> <ul style="list-style-type: none"> (k) a printed log, (l) a card file, (m) a system file, (n) a user file. <p>(o-p) The system must allow job charge information to be displayed at a:</p> <ul style="list-style-type: none"> (o) users terminal, (p) central site device | <p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>6</p> | | |

3.3.1 MAINTAINING JOB CHARGE INFORMATION

Reference:

1. Many of these criteria are highly desirable for a batch or time-sharing processing system in which job charge information is required. This type of information is also useful in evaluating system utilization.
2. This criterion is highly desirable for all processing systems that require accounting routines, since each installation usually has a need for separate charge calculations.
3. This function frequently occurs in a batch processing system.
4. This method is rarely used and should only be considered for small systems with insufficient IAS.
5. This criterion should be specified if the accounting routines are system supplied.
6. This criterion should be specified if the accounting routines are supplied by the installation.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 3.3.2 MAINTAINING ERROR STATISTICS | | | |
| (a) The system must accumulate information for a hardware error summary. | 1 | | |
| (b) The system must accumulate information for a program error summary. | 2 | | |
| (c) The system must provide facilities for the analysis of error statistics (e.g., indication of faulty I/O devices, etc.). | 3 | | |
| (d) The system must provide lists of file access violation attempts. | 4 | | |
| 3.3.3 MAINTAINING SYSTEM UTILIZATION STATISTICS | | | |
| (a) The system must maintain a summary by user account. | 1 | | |
| (b) The system must maintain a summary of file accesses. | 1 | | |
| (c) The system must maintain a summary of system service requests. | | | |
| (d) The system must provide performance monitoring information. | 2 | | |

3.3.2 MAINTAINING ERROR STATISTICS

Reference:

1. This criterion should be considered for specification for all processing system types as an aid to personnel performing hardware diagnostic maintenance.
2. This function occurs infrequently, and then is usually restricted to batch processing systems as an aid in determining which programs are continually incurring error conditions.
3. This criterion should be specified for all processing systems that accumulate a hardware error summary.
4. This criterion is highly desirable for all processing systems that provide file access security controls.

3.3.3 MAINTAINING SYSTEM UTILIZATION STATISTICS

Reference:

1. These criteria should be considered for specification in multi-user batch processing and time-sharing systems.
2. These criteria are desirable when a general purpose system is to be tailored to a specific installation. The information provided should present sufficient details to enable the installation to modify system generation parameters for more efficient system operation.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-------------|--------------|----------|
| | | Initial | Extended |
| <p>3.4.1. CURRENT SYSTEM STATUS INTERROGATION</p> <p>(a-h) The system must maintain the following status information.</p> <ul style="list-style-type: none"> (a) number of jobs, (b) number of interactive users, (c) list of active terminals, (d) main storage allocation (e) secondary storage allocation, (f) device allocation, (g) device status, (h) elapsed execution time. | 1 | | |
| <p>3.4.2 SYSTEM DEFINITION INTERROGATION</p> <p>(a-c) The system must maintain the following definition information:</p> <ul style="list-style-type: none"> (a) hardware configuration, (b) software configuration, (c) system limits (e.g., number of jobs allowed, etc.). | 1 1 2 | | |

3.4.1 CURRENT SYSTEM STATUS INTERROGATION

Reference:

1. These criteria should be considered for specification since they can be useful to a facility in monitoring system operation. These system features can also be useful during system evaluation.

3.4.2 SYSTEM DEFINITION INTERROGATION

Reference:

1. This criterion is highly desirable in all processing systems in which adaptive programs are available e.g., a compiler that can operate with various configurations or an application program that can use various devices or reside in various core sizes.
2. This criterion is highly desirable in any system in which performance monitoring is to be performed.

3.2.4 Fourth Level of Detail (Part I - Executive/Control Functions)

This subsection covers the following fourth level executive/control functions:

- 1.1.1.1 ALGORITHMIC SCHEDULING
- 1.1.1.2 TIME INITIATED SCHEDULING
- 1.1.1.3 EVENT INITIATED SCHEDULING
- 1.1.1.4 PROGRAM INITIATED SCHEDULING
- 1.1.1.5 CONDITIONAL SCHEDULING
- 1.1.1.6 SCHEDULING QUEUE MAINTENANCE
- 1.1.2.1 CORE STORAGE ALLOCATION
- 1.1.2.2 I/O DEVICE ALLOCATION
- 1.1.2.3 COMMON ROUTINE ALLOCATION
- 1.1.3.1 LOADING CONTROL
- 1.1.3.2 SWAPPING CONTROL
- 1.1.3.3 STRUCTURE CONTROL
- 1.1.4.1 DISPATCHING CONTROL
- 1.1.4.2 EVENT SYNCHRONIZATION
- 1.1.4.3 INTERRUPT PROCESSING CONTROL
- 1.1.4.4 PROGRAM LIMIT MONITORING
- 1.2.2.1 BUFFERING CONTROL
- 1.2.2.2 DATA CODE TRANSLATION
- 1.3.1.1 SYSTEM INITIALIZATION
- 1.3.1.2 SYSTEM RESTART
- 1.4.2.1 PROGRAM INITIATED RESTARTING
- 1.4.2.2 SYSTEM INITIATED RESTARTING
- 1.4.2.3 DEVICE REPOSITIONING

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.1.1 ALGORITHMIC SCHEDULING | 1 | | |
| (a) The scheduling algorithm must recognize job priority. | | | |
| (b) The scheduling algorithm must ascertain resource availability prior to job initiation. | 2 | | |
| (c) The scheduling algorithm must be responsive to the length of time a program has remained unscheduled in a scheduling queue. | 3 | | |
| (d) The scheduling algorithm must take into consideration estimated time requirements. | | | |
| (e) The scheduling algorithm must take into consideration predictions as to whether a given job will be I/O or CPU bound. | 4 | | |
| (f) The system must permit operator modification of job priorities. | | | |
| (g) The system must permit operator modification of the scheduling algorithm. | | | |

1.1.1.1 ALGORITHMIC SCHEDULING

Reference:

1. Algorithmic scheduling is a priority scheduling concept where many variables influence the selection of the program chosen for execution.
2. It is generally desirable that all the resources a program may need be made available prior to scheduling the program for execution. For example, if this were not done and a named input file was not on-line when the program was scheduled, the program would be unable to execute until the file was located, mounted, and recognized by the system. However, the program would still be occupying core storage which could otherwise be used.
3. Since it is possible, because of low priority or lack of resources, for a program to be delayed in the scheduling queue for a long time, this variable is sometimes included within the scheduling algorithm. If a program has been waiting over an established time limit the system can begin to reserve the resources that it requires as they become free in order to insure that all required resources will be available.
4. A parameter denoting those programs which are either I/O or CPU bound can enable the scheduler to develop a better job mix to maximize system utilization and/or throughput.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.1.2 TIME INITIATED SCHEDULING | | | |
| (a) The system must permit a job to be scheduled at a specified time of day. | 1 | | |
| (b) The system must permit a job to be scheduled after a specified time interval. | 2 | | |
| (c) The system must permit a job to be scheduled periodically at each elapsement of a specified time interval. | 2 | | |
| (d) The system must permit specification of a time deadline for scheduling a job. | 1 | | |
| 1.1.1.3 EVENT INITIATED SCHEDULING | | | |
| (a) The system must provide scheduling based upon different event priority levels. | 1 | | |
| (b-h) The system must provide scheduling based upon the following types of events: | | | |
| (b) communications interrupts, | 2 | | |
| (c) operator interrupts, | 3 | | |
| (d) process control interrupts, | | | |
| (e) error conditions, | 3 | | |
| (f) I/O completion, | | | |
| (g) task completion, | | | |
| (h) unsolicited key-ins. | 2 | | |

1.1.1.2 TIME INITIATED SCHEDULING

Reference:

1. This criterion should be considered for real-time and batch processing applications.
2. This criterion should be considered for real-time processing systems that monitor input from external devices either on a periodic or elapsed interval basis e.g., teleprocessing, telemetry updates, analog updates, etc.

1.1.1.3 EVENT INITIATED SCHEDULING

Reference:

1. Multiple priority levels will most likely be required in time-critical environments.
2. This criterion should be specified for real-time, remote batch and interactive processing systems.
3. This criterion should be specified for all processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|--|--------------|----------|
| | | Initial | Extended |
| <p>1.1.1.4 PROGRAM INITIATED SCHEDULING</p> <p>(a-c) The system must provide for program initiated scheduling of:</p> <p>(a) symbionts,</p> <p>(b) subprograms/tasks,</p> <p>(c) independent programs/tasks.</p> <p>(d) The system must provide scheduling for immediate execution.</p> <p>(e) The system must provide scheduling for asynchronous execution.</p> <p>(f) The system must provide scheduling for subsequent execution.</p> | <p>1</p> <p>2</p> <p>3</p> <p>3</p> <p>3</p> | | |

1.1.1.4 PROGRAM INITIATED SCHEDULING

Reference:

1. This feature is most frequently found in large multiprogrammed batch processing systems.
2. This feature is frequently found in time-sharing systems where a foreground time-sharing program initiates a batch processing background program.
3. The two simplest methods of program initiated scheduling are immediate execution and subsequent execution. In the immediate method the program in execution initiates a call for another program to be scheduled. The calling program is suspended until the called program terminates. When subsequent execution is specified, the called program is not executed until the calling program terminates.

The most difficult but most versatile method is asynchronous execution whereby both the calling program and the called program execute concurrently.

1.1.1.5 CONDITIONAL SCHEDULING

Reference:

1. While frequently employed for evaluation, these criteria are rarely specified. The most likely reason for specification would be when designing a special purpose batch processing system for a specific application. The intended application program capabilities and design should have been analyzed in sufficient detail to justify the need for this specification.

1.1.1.6 SCHEDULING QUEUE MAINTENANCE

Reference:

1. These criteria are highly dependent upon the hardware configuration and can only be specified in detail when the hardware configuration is known.
2. The number of job entries in each queue should be based on the anticipated work load and the degree of job stacking that will occur. For example, a remote terminal that submitted all of its batch jobs at a single time rather than on a random basis would require a relatively large queue. The amount of secondary storage that will be available for job stacking must also be taken into account.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|---|--------------|----------|
| | | Initial | Extended |
| <p>1.1.2.1 CORE STORAGE ALLOCATION</p> <p>(a-d) Static core storage allocation will be provided at the following levels:</p> <p>(a) partition,</p> <p>(b) job,</p> <p>(c) job step,</p> <p>(d) task.</p> <p>(e-i) Core storage must remain allocated until:</p> <p>(e) explicit deallocation,</p> <p>(f) job completion,</p> <p>(g) job step completion,</p> <p>(h) task completion,</p> <p>(i) abnormal termination</p> <p>(j-m) The system must provide static (fixed) core allocation for:</p> <p>(j) program expansion,</p> <p>(k) I/O buffers,</p> <p>(l) common areas,</p> <p>(m) subtask execution.</p> <p>(n-q) The system must provide dynamic core allocation for:</p> <p>(n) program expansion,</p> <p>(o) I/O buffers,</p> <p>(p) common areas,</p> <p>(q) subtask execution.</p> <p>(r) The system must permit common (shared) core allocation between tasks of the same job step.</p> <p>(s) The system must permit common core allocation between job steps of a job.</p> <p>(t) The system must permit common core allocation between jobs.</p> | <p></p> <p></p> <p>1</p> <p>2</p> <p>2</p> <p>2</p> <p>3</p> <p>4</p> <p>4</p> <p>5</p> <p>5</p> <p>6</p> | | |

1.1.2.1 CORE STORAGE ALLOCATION

Reference:

1. This method is generally used in most basic multiprogramming systems and in small to medium scale extended multiprogramming systems. It may also be found in systems which support concurrent operation of different environments e.g., real-time processing in a foreground partition and batch processing in a background partition.
2. These methods of allocation occur in most time-sharing systems as well as in medium to large scale extended multiprogramming systems.
3. If core is allocated at the partition level, this criterion is unnecessary.
4. The previous level of criteria specification (1.1.2) will have determined whether a static, dynamic, or combined method of core allocation will be employed. In certain instances it becomes necessary for a program to expand above core estimates during operation due to a particular data test or tested option. Core for I/O buffers is required in order to efficiently transfer data or information between core and external devices without continually interfering with program execution. Core for common areas is used when there is more than one program dependent upon previously acquired data or information. Core for subtask execution is derived from the possible scheduling of a subtask by a task in execution.
5. This criterion is highly desirable for most batch processing systems.
6. This criteria should only be specified if it has been determined that it is desirable for multiple jobs to communicate and pass data to each other. This solution avoids the problem of passing data via secondary storage files. It is most frequently used in teleprocessing applications where the message reader and message writer pass and receive data from the processing program via a core buffer rather than secondary storage areas.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.2.1 CORE STORAGE ALLOCATION (cont'd.) | | | |
| (u) The system must provide storage protection against unauthorized program access. | 7 | | |
| (v) The system must provide storage write protection. | 7 | | |
| (w) The system must provide storage read protection. | 7 | | |

1.1.2.1 CORE STORAGE ALLOCATION (cont'd.)

Reference:

7. This criterion should be specified for all multiprogramming processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | | REQUIREMENTS | |
|--|---|--------------|----------|
| | | Initial | Extended |
| 1.1.2.2 I/O DEVICE ALLOCATION | | | |
| (a-b) The system must provide the following types of allocation: | 1 | | |
| (a) physical I/O devices, | 2 | | |
| (b) logical I/O files | | | |
| (c-f) The system must permit device specification at the following levels: | | | |
| (c) physical device reference, | 3 | | |
| (d) symbolic device reference, | 4 | | |
| (e) generic device reference (e.g., tape, disk, etc.), | 5 | | |
| (f) access method reference (e.g., sequential, direct access). | 6 | | |
| (g) The system must provide exclusive allocation of devices/files. | | | |
| (h) The system must provide shared allocation of devices/files. | 7 | | |
| (i-m) I/O devices must remain allocated until: | | | |
| (i) explicit deallocation, | 8 | | |
| (j) job completion, | | | |
| (k) job step completion, | | | |
| (l) task completion, | | | |
| (m) abnormal termination. | 8 | | |

1.1.2.2 I/O DEVICE ALLOCATION

Reference:

1. It is generally advisable for a multiprogrammed system to control the allocation of physical I/O devices and logical I/O files. In this way, better device utilization may be achieved by assigning unused devices to any program requiring one. The system controlled allocation of logical I/O files is necessary if file security access controls are to be implemented.
2. These are, in order, increasingly preferred methods of allocating I/O devices in a multiprogrammed environment.
3. This method is frequently used in real-time and communication processing systems.
4. Symbolic device references are particularly desirable when device substitution and reconfiguration features are also implemented.
5. Generic device reference allows the system to assign any device within a device group to a program for usage e.g., any tape drive, any disk, etc. This technique greatly increases system utilization because the OS can assign devices that it knows are not in use.
6. The access method reference technique is considered one step above generic reference in that the program merely states the requirement for either a sequential or direct access device and the system does the rest. This method of device reference allows greater system control of devices and greater utilization of devices.
7. This method is desirable in most time-sharing and extended multi-programming systems.
8. This criterion should be specified if dynamic allocation is specified.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|----------------------|--------------|----------|
| | | Initial | Extended |
| 1.1.2.3 COMMON ROUTINE ALLOCATION | | | |
| (a-c) The system must support the following routine/module types: (a) non-reusable, (b) serially reusable, (c) reentrant. | 1 | | |
| (d-g) The system will allow common routines to reside in the following areas: (d) transient execution area, (e) permanent execution area within the supervisor, (f) problem program area, (g) independent partition. | 2 3 4 4 | | |

1.1.2.3 COMMON ROUTINE ALLOCATION

Reference:

1. Programs or subroutines that can be used by more than one program are frequently designed to be loaded and executed when needed rather than incorporated as a part of the executing program during the binding process. These routines may be loaded into a specially-designated transient execution area, into any available area of core the executing program can find, or, if used frequently enough, made a part of the resident system. Allocation of these routines to requests is almost always dynamic rather than static.

There are three types of subroutine organization that require different allocation procedures. Non-reusable subroutines are subroutines that must be loaded "fresh" each time a request is made for the routine. Serially re-usable routines need not be reloaded; however, the routine, because of possible modification of constants or instructions, can only be used by one program at a time. Re-entrant routines may be used by any number of executing programs simultaneously. Thus, allocation of re-entrant routines is straightforward, whereas serially reusable routines must have a lock/unlock facility which limits their assignment to a single program at a time.

2. While frequently employed for evaluation, these criteria are rarely specified.
3. A transient execution area is an area set aside to hold one or more routines that normally reside on secondary storage and which are loaded only when actually required. This technique is frequently employed for executive control routines with low utilization rates to provide more available main storage for application programs. The technique may, however, also be used by application programs to minimize their main storage requirements.
4. This method frequently occurs in multiprogramming systems for common routines which have a high utilization rate. Although this method requires permanent core, it is offset by fewer requests for transient routines.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.3.1 LOADING CONTROL | | | |
| (a-c) The system must allow loading to be initiated by: | | | |
| (a) control statement reference, | 1 | | |
| (b) explicit program reference, | 2 | | |
| (c) implicit program reference. | 3 | | |
| (d-f) The system must provide facilities for the compaction of fragmented core, to be initiated: | 4 | | |
| (d) at termination of each task, | | | |
| (e) when dictated by priority requirements, | | | |
| (f) when directed by the operator, | | | |
| (g) periodically by the system. | | | |
| (h) The system must provide automatic overlay loading. | 5 | | |
| (i) The system must provide directed overlay loading. | 5 | | |

1.1.3.1 LOADING CONTROL

Reference

1. This criterion should be specified for a batch processing system.
2. This method is frequently used in processing systems that support overlay loading. In this way the overlay segment in residence initiates the loading of the next overlay segment.
3. This technique is generally used with segmented or paged program structures. If an executing program tries to transfer control to or access data from a non-resident segment or page, an interrupt is generated, the referenced segment or page is loaded, and the instruction is re-issued.
4. While frequently employed as an evaluation measure this criterion is rarely specified. However, if a multiprogrammed system is to utilize core to the maximum extent possible compaction is a requirement and should be considered for both specification and evaluation. If the system supports a paged mode of memory allocation, however, compaction is generally unnecessary.

Compaction at the termination of each task permits maximum utilization of core but increases system overhead tremendously. Compaction dictated by priority requirements will minimize compaction overhead while making it more likely that a priority program can operate without causing the termination of other programs. Operator directed compaction is better than none at all; however, it tends to be sporadic and may not increase core utilization to a significant extent. Some systems, to minimize overhead, only initiate core compaction periodically on a time or program termination count interval.

5. This criterion should be considered for specification if the system is to use overlays. In automatic overlay loading this function is entirely transparent to the programmer and can be likened to a virtual memory concept. In directed overlay loading the overlay in residence must initiate the loading of the next overlay and this load initiation is prepared externally and becomes part of the program structure.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.3.2 SWAPPING CONTROL | 1 | | |
| (a) The system must allow temporary program expansion into an additional area of core by rolling-out lower priority programs. | 2 | | |
| (b) The system must allow programs to time share core storage. | 3 | | |

1.1.3.2 SWAPPING CONTROL

Reference:

1. Swapping is a method of sharing main storage between several programs by maintaining each program and its status on secondary storage and loading each one into main storage for a limited time interval.

Certain systems use paging as a swapping technique in which pre-defined program segments (pages) are swapped in and out of core. This is rarely specified unless the system hardware configuration is such that most programs will require loading or swapping from IAS in which case paging should be considered for specification. If paging is specified then it should also be specified that the OS will allow multiple pages with the page size entirely dependent upon the architecture of the IAS device to be used.

2. This method is frequently used in processing systems that support concurrent modes of operation e.g., real-time foreground and batch background.
3. This function frequently occurs in small to medium size time-sharing processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.3.3 STRUCTURE CONTROL | 1 | | |
| (a-d) The system must support the following types of program structure: <ul style="list-style-type: none"> (a) simple, (b) overlay, (c) paged, (d) dynamic. | 2 | | |

1.1.3.3 STRUCTURE CONTROL

Reference:

1. These criteria are generally employed for evaluation, rather than specification. In some isolated circumstances, however, it may be necessary to specify the program structuring technique.
2. A simple program structure consists of a single module occupying a fixed amount of main storage. An overlay program allows for repeated use of the same blocks of core storage during different stages of program execution. Thus, the overlay program is segmented and each segment is loaded as it is required.

Paged and dynamic structures are considerably more sophisticated in scope and are common only in the larger multiprogrammed and time-sharing environments. Dynamic loading allows a relocatable module to be loaded into main storage upon a request for that module by an executing program. It differs from an overlay structure in that the area of main storage need not be prespecified, nor does the module necessarily replace or overlay an existing program segment. One of the major problems with dynamically-loaded programs is that the process creates unused core fragments when all dynamically-loaded programs do not terminate concurrently. Consequently, some systems relocate all active program segments to a contiguous core storage area prior to dynamically loading another module. Others only compact storage when core becomes exhausted or when certain high priority programs require loading.

When storage is divided into pages the program loading function may consist of simply loading all required main storage pages, or it may entail preparing the system for a process called paging. Paging requires a fast access secondary storage device (usually a magnetic drum) as a supplement to main storage. Both the secondary storage device and main storage are configured to the same fixed page size. A program resides on both devices; the page of the program containing the instruction sequence currently executing is in main storage, many of the other pages are in secondary storage. Whenever a page not in main storage is referenced for data or to transfer control, a page area is made ready in main storage -- perhaps by moving an in-core page to secondary storage -- and the required page is read in. A page map is maintained indicating the physical page assignments (main storage or secondary storage) and the actual storage address (core locations or track). This map is consulted to resolve addresses when accessing the data or instructions within the page. In many systems, this address resolution is accomplished through hardware circuitry; in others, it must be accomplished via software.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-------------------|--------------|----------|
| | | Initial | Extended |
| <p>1.1.4.1 DISPATCHING CONTROL</p> <p>(a) The dispatching algorithm must be modifiable at system generation time.</p> <p>(b) The dispatching algorithm must be modifiable by the operator.</p> <p>(c) The system must allow up to _____ entries in each dispatch queue.</p> | <p>1</p> <p>2</p> | | |
| <p>1.1.4.2 EVENT SYNCHRONIZATION</p> <p>(a-d) The system must recognize the following events:</p> <p>(a) I/O completion,</p> <p>(b) time interval timeout,</p> <p>(c) program completion/abnormal termination,</p> <p>(d) unsolicited key-in.</p> | <p>1</p> | | |

1.1.4.1 DISPATCHING CONTROL

Reference:

1. The dispatching algorithm is the heart of every multiprogramming and time-sharing system. This algorithm controls the allocation of processor time to each contending task and is, with the scheduling algorithm, the factor determining single job throughput. Modifications to the algorithm, while desirable, should only be incorporated after a comprehensive analysis of system performance.
2. In basic multiprogramming the number of entries in the dispatch queue is equal to the number of partitions.

In extended multiprogramming the number of entries in the dispatch queue is equal to the number of possible core resident jobs.

In a time-sharing processing system the number of entries in the queue is equal to the number of users.

1.1.4.2 EVENT SYNCHRONIZATION

Reference:

1. Event synchronization is the process of delaying task execution until some specified event occurs or the process of triggering a task upon the occurrence of a specified event. The most common types of events which may delay or initiate task execution are I/O completions, selected time intervals, subtask completions, and unsolicited key-ins.

Event synchronization may also be effected between several tasks of a job. One task may initiate any number of subordinate tasks and may execute concurrently with them. This main task may, at any time, issue wait requests which will suspend main task processing until one or more of the subordinate tasks have been completed.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| <p>1.1.4.3 INTERRUPT PROCESSING CONTROL</p> <p>(a-d) The system must provide the following interrupt processing options:</p> <ul style="list-style-type: none"> (a) priority recognition, (b) interrupt masking, (c) interrupt disarming, (d) stacked interrupt processing. | 1 | | |

1.1.4.3 INTERRUPT PROCESSING CONTROL

Reference:

1. An interrupt, as the name indicates, is a break in the normal flow of program execution. An interrupt is usually caused by a hardware-generated signal such as an I/O event, a program error, a machine error, or an operator-initiated signal.

In a real-time system, the interrupt levels of the various events to be monitored are usually specified when the system is generated; in general purpose systems, they are usually intrinsic to the system and may not be specified. Normally, processing on a given level will be interrupted whenever higher level interrupts occur; interrupts of a lower level than the processing program will be held pending. If several interrupts occur simultaneously, they are stacked and honored according to their priority. Unfortunately, in some systems inadequate stack lengths can cause multiple interrupt signals to be lost.

Interrupt acknowledgement can be altered by disabling interrupts to prevent their recognition or by masking them to defer their recognition. When an unmasked interrupt occurs, the current instruction is usually allowed to complete execution. Then the program and the contents of the machine registers are saved in main storage and control is transferred to a program which will service the interrupt. This program may complete execution or in turn may be interrupted by a still higher priority interrupt. When an interrupt processing program does complete execution, control is returned to the highest pending interrupt processing routine or to the supervisor dispatching routine if no interrupts are pending.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| <p>1.1.4.4 PROGRAM LIMIT MONITORING</p> <p>(a-g) The system must permit specification of limits for:</p> <ul style="list-style-type: none"> (a) execution time, (b) number of input records, (c-e) number of output: <ul style="list-style-type: none"> (c) lines, (d) cards, (e) records, (f) main storage utilization, (g) secondary storage utilization. | 1 | | |

1.1.4.4 PROGRAM LIMIT MONITORING

Reference:

1. Many systems monitor various resources during job execution to insure that certain specified limits are not exceeded. Limits are usually established for such elements as central processor time used, output records produced, and the amount of main and secondary storage space allocated. While installation maximums for each resource are normally established at system generation time, they can usually be overridden by job control cards. Some systems automatically cause abnormal job termination when any of the system limits for a particular job are exceeded. Other systems permit the user to specify other actions to be exercised, such as operator or program notification, whenever the various limits are exceeded.

In some smaller real-time systems, execution time limits are not explicitly set; however, the supervisor maintains a countdown loop which will time out unless it is periodically reset by the executing program. If the countdown loop times out, an interrupt is generated and program execution terminates.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.2.1 BUFFERING CONTROL | 1 | | |
| (a) The system buffering concept must be based upon user specified areas. | 2 | | |
| (b) The system must provide system buffer pools. | 3 | | |
| (c) The system must permit user buffer pools. | 3 | | |
| (d-f) The system must provide the following buffering methods: (d) simple buffering, (e) exchange buffering, (f) chained segment buffering. | | | |
| (g) The system must permit static program specification of buffer areas. | 4 | | |
| (h) The system must allow buffer assignment via control statements. | 4 | | |
| (i) The system must permit dynamic program specification of buffer areas. | 4 | | |

1.2.2.1 BUFFERING CONTROL

Reference:

1. While frequently employed for evaluation this criterion is rarely specified.
2. This method is frequently used in a serial processing system or in a small real-time processing system in which the user performs the I/O operations.

Buffering based upon user specified areas is the most easily implemented method; however, this does require that areas of core be set aside for the duration of the program.

3. An alternate method of approaching the buffer problem is the use of buffer pools. This provides an area of core that will always be used for buffering. There is not much difference between system buffer pools and user buffer pools except by using system buffer pools the user is relieved of pool handling problems. Some systems employ both user and system buffer pools.
4. Static program specification of buffer areas means that the buffer areas are set aside during the entire execution of the program. Since the buffers are statically assigned there is no flexibility in altering the assignment other than a program change. This technique is generally prevalent in small scale systems.

Control statement specification of buffer areas means that the buffer will not be assigned until the control statement is executed, however, it will remain assigned throughout program execution. This is a flexible means of assigning buffers since the control statement is modifiable input. This technique is generally prevalent in medium scale systems.

Dynamic program specification is one of the most advanced methods of assigning buffer areas since the buffer areas are only assigned as the program requires them and can usually be released by the program when no longer needed. This technique is highly desirable in large scale systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.2.2 DATA CODE TRANSLATION | | | |
| (a) The system must provide data compaction/expansion facilities. | 1 | | |
| (b) The system must provide character code conversion facilities. | 2 | | |
| (c) The system must provide device code conversion facilities (e.g., paper tape codes, teleprocessing codes). | 2 | | |

1.2.2.2 DATA CODE TRANSLATION

Reference:

1. This criterion should be considered for specification for all processing systems that will be processing large amounts of data, since this method will decrease storage and transmission requirements.
2. This criterion is highly dependent upon the types of hardware to be used, however, it should be considered for all processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.1.1 SYSTEM INITIALIZATION | | | |
| (a) The system must permit modification of partition sizes. | 1 | | |
| (b) The system must permit assignment/modification of partition priorities. | 1 | | |
| (c) The system must permit assignment/modification of time-slicing specifications. | 2 | | |
| 1.3.1.2 SYSTEM RESTART | | | |
| (a) The system must schedule installation restart programs. | 1 | | |
| (b) The system must automatically restart jobs which were in a state of execution at the time the system came to a halt. | 2 | | |
| (c) The system must automatically reschedule queued jobs. | 3 | | |
| (d) Job queues must be restored by the system to their condition prior to the system halt. | 3 | | |
| (e) The system must provide manual reconfiguration procedures. | 4 | | |
| (f) The system must provide automatic reconfiguration procedures. | 4 | | |

1.3.1.1 SYSTEM INITIALIZATION

Reference:

1. This criterion should be specified if a system is to provide fixed partitions since it will provide flexibility in meeting daily requirements.
2. This feature is desirable if subsequent tailoring of a time-sharing or multiprogramming system is anticipated.

1.3.1.2 SYSTEM RESTART

Reference:

1. This feature frequently occurs in a real-time processing system in which unique procedures must be performed during restart. This method is also somewhat useful in supporting batch processing environments.
2. This criterion is highly desirable for all processing systems though its implementation is somewhat costly. Usually in a real-time monitor system automatic restart of jobs after a system halt is imperative.
3. This function frequently occurs in extended multiprogramming systems. Normally information relative to job queues will not be affected by a system halt.
4. This criterion is a highly desirable feature for all processing systems since it allows continued operation despite failing devices. While both manual and automatic reconfiguration are prevalent, automatic techniques are normally required to support real-time processing due to the speed in which it is performed. Manual configuration can be quite appropriate for batch and time-sharing environments.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|------------|--------------|----------|
| | | Initial | Extended |
| <p>1.4.2.1 PROGRAM INITIATED RESTARTING</p> <p>(a) The system must require operator consent before allowing a program initiated restart.</p> <p>(b) The system must allow for the optional specification of operator consent before allowing a program initiated restart.</p> | 1 | | |
| <p>1.4.2.2 SYSTEM INITIATED RESTARTING</p> <p>(a-c) The system must provide for restart under the following conditions:</p> <p>(a) device error recovery,</p> <p>(b) system error recovery,</p> <p>(c) power failure.</p> | 1 2 | | |

1.4.2.1 PROGRAM INITIATED RESTARTING

Reference:

1. This criterion is highly desirable for real-time processing systems. When a real-time program experiences an error, reliability factors can usually determine whether operator consent is required prior to restart.

1.4.2.2 SYSTEM INITIATED RESTARTING

Reference:

1. These criteria are highly desirable for a real-time processing system due to the usual requirement for continuous support. These criteria can also be quite beneficial for batch and time-sharing processing systems as a means of avoiding total system re-initialization.
2. The capability of the OS to support this requirement is quite dependent upon a system hardware's non-destructive memory capability.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.4.2.3 DEVICE REPOSITIONING | | | |
| (a) The system must re-establish teleprocessing line connections. | 1 | | |
| (b) The system must retransmit interrupted telecommunications messages. | 2 | | |
| (c) The system must reposition temporary files. | 3 | | |
| (d) The system must reposition permanent files. | 3 | | |
| (e-f) The system must reposition the following devices: | | | |
| (e) sequential access devices, | | | |
| (f) direct access devices. | 4 | | |

1.4.2.3 DEVICE REPOSITIONING

Reference:

1. This criterion is highly desirable for systems supporting teleprocessing. Usually it is better to have the system perform this function than the operator since the connection can be derived from a predefined procedural matrix and can be performed much faster and accurately by the system.
2. This criterion should be specified for all systems supporting teleprocessing to assure that no messages are lost due to system failure.
3. This criterion should be considered for systems containing files on sequential access devices.
4. The specification of this criterion is usually not required except in situations where direct access device commands do not perform an automatic positioning function.

3.3 Requirements Checklist - Part II - System Management Functions

The following subsections present specification and evaluation criteria for the non-real-time components of the operating system which support and maintain both system and application programs.

3.3.1 First Level of Detail (Part II - System Management Functions)

This subsection covers the following first level system management functions:

- 1.0 OPERATING SYSTEM MANAGEMENT
- 2.0 PROGRAM MAINTENANCE
- 3.0 COMPILER INTERFACES
- 4.0 MANAGEMENT SUPPORT UTILITIES

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.0 OPERATING SYSTEM MANAGEMENT | 1 | | |
| (a) System generation must be provided. | 2,4 | | |
| (b) System maintenance must be provided. | 3,4 | | |

1.0 OPERATING SYSTEM MANAGEMENT

Reference:

1. Operating system management routines are provided by all systems to enable each installation to generate and maintain a version of the computer operating system.
2. System generation is the process of tailoring and adapting a generalized operating system to the specific machine configuration and operational requirements of an installation. The generalized master operating system is normally provided by the vendor to every installation. The master system contains all the operating system routines required for any allowable system configuration as well as the vendor-developed optional routines that might be desired by a particular installation.
3. System maintenance allows an installation to update the operating system in response to changes in the operating environment or to changes of the programs within the operating system itself. The latter category is generally related to vendor-supplied updates which are distributed periodically. Consequently, this type of maintenance tends to be performed fairly regularly and generally necessitates suspension of all other processing activity until the update is complete. Extensive updates to the system may involve a total regeneration of the system; however, minor changes may frequently be effected by simply replacing selected system modules.
4. This criterion should be specified for all processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.0 PROGRAM MAINTENANCE | 1 | | |
| (a) The system must provide facilities for program library maintenance. | 2 | | |
| (b) The system must provide maintenance facilities for system control card libraries. | 3 | | |
| (c) The system must provide facilities for generating executable program modules. | 4 | | |

2.0 PROGRAM MAINTENANCE

Reference:

1. Program maintenance facilities support the maintenance and modification of application programs within the framework of the system. These facilities are distinct from the support features that an application program can call upon when executing; rather these facilities treat the application program as a product by itself.
2. Many systems provide capabilities for maintaining one or more program types in indexed libraries. In general, a program library is a collection of routines or instruction sets which are all represented at the same code level. The code levels generally found in contemporary operating system libraries are: macro code, source program code, relocatable program code, executable code, and job control procedures.
3. This function is highly desirable for a batch processing system. When a great number of system control cards are required for the execution of a job, it is best for the system to catalog these control cards into a named sequence. This named sequence can then be referenced by one control card in the input stream rather than continually inserting multiple control cards in the input stream.
4. This criterion should be specified for all processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 3.0 COMPILER INTERFACES | 1 | | |
| (a-c) The system must provide the following executive routine support for compilers: | 2 | | |
| (a) recognition of compiler parameters on OS control cards, | | | |
| (b) system maintained compiler communication tables, | | | |
| (c) program testing/debugging control. | | | |
| (d-f) The system must provide the following types of libraries in support of compilers: | 3 | | |
| (d) source program libraries, | | | |
| (e) macro libraries, | | | |
| (f) subroutine libraries. | | | |
| (g-i) The system must provide the following system utilities in support of compiler language programs: | 4 | | |
| (g) sort/merge programs, | | | |
| (h) peripheral conversion programs, | | | |
| (i) data management system programs. | | | |

3.0 COMPILER INTERFACES

Reference:

1. Those operating system functions that provide supporting services to the system language compilers are the least standard of all operating system functional categories. Furthermore, differences can not be directly attributed to the size, orientation, or sophistication of the operating system.
2. Many operating systems grant the system compilers selected privileges which are not available to other non-supervisory programs. For example, several systems allow the compiler to use communication tables within the resident executive for passing parameters and storing specifications about the compilation. Many systems also provide special job control cards for compilers which include compilation parameters along with normal operating system parameters. In some cases the executive system may actually decode these parameters and place them in an appropriate compiler communication table.

Other types of executive routine support are found in those systems that recognize uniquely formatted compiler input/output files and provide non-standard input and output symbionts for processing them. Finally, some systems also provide a series of compilation error codes which can be used as conditional scheduling parameters by subsequent tasks and steps within the job.

3. A few systems provide and maintain unique libraries for the exclusive use of the system compilers. These libraries may take the form of compiler source program libraries, macro statement libraries, or compiler subroutine libraries. In general, the same maintenance facilities that are available for other system libraries are available for compiler-oriented libraries.
4. Some compilers have the capability of generating linkage sequences to selected system utility programs in order that these facilities can become available to the compiler language programmer. For example, COBOL compilers commonly allow references to the system sort and merge functions. Other utility functions that can be invoked in some systems are the data management and data handling support functions.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 4.0 MANAGEMENT SUPPORT UTILITIES | 1 | | |
| (a) The system must provide facilities for peripheral device support. | 2 | | |
| (b) The system must provide facilities for simulation routines. | 3 | | |
| (c) The system must provide facilities for performance measurement routines. | 4 | | |
| (d) The system must provide facilities for stand-alone utilities. | 5 | | |

4.0 MANAGEMENT SUPPORT UTILITIES

Reference:

1. The management support utilities are primarily for the use of the system manager rather than the average user. Utility functions provided for normal users are listed in Part III, Data Manipulation Functions, Section 2.0.
2. This function should be considered for specification for all processing systems.
3. This function is highly desirable in a real-time processing system or a system that supports teleprocessing oriented programs. Certain real-time programs can only be invoked by the occurrence of specific interrupts. It may be difficult to provide the actual interrupt when these programs must be tested. Consequently, a capability to simulate the occurrence of the interrupts and to thereby invoke the program permits much greater flexibility in program testing.

Similarly, an exhaustive test of communication and line support programs might require a scenario so complex that it could not be performed from available support terminals. Again, the capability to simulate the arrival of a series of communication messages provides a needed testing capability. Many of these routines might also be used by the system manager to test and validate the operating system itself.

4. Systems measurement routines enable the system manager to obtain various statistics about the operational use of the system. Typical statistics include job through-put times, file or device utilization figures, and the identification of bottleneck conditions. In addition, there are a few systems that provide visual displays of current and past system utilization reflecting error statistics on all configured devices, channel usage, memory allocation, programs in core, programs swapped out of core, and processor time consumed by each program at that point in time.
5. These routines are primarily used when the system has failed and normal diagnostic routines are incapable of restarting the system. Two types of routines are provided. The first type includes those status display routines which produce core dumps, file dumps, and dumps of selected diagnostic log-out areas. These dumps are presented for interpretation by a hardware or software engineer to determine the cause of the failure.

The second type consists of recovery support routines which enable the system to reconstruct much of its pre-failure environment and to reinitiate processing by rebuilding selected queues, reloading various programs, and restarting check-pointed programs. The recovery support functions are most often found in systems oriented to supporting real-time or communication-based processing.

3.3.2 Second Level of Detail (Part II - System Management Functions)

This subsection covers the following second level system management functions:

- 1.1 SYSTEM GENERATION
- 1.2 SYSTEM MAINTENANCE
- 2.1 LIBRARY AND DIRECTORY MAINTENANCE
- 2.2 LOAD MODULE GENERATION
- 4.1 PERIPHERAL DEVICE SUPPORT
- 4.2 SYSTEM SIMULATION ROUTINES
- 4.3 SYSTEM MEASUREMENT ROUTINES
- 4.4 STAND-ALONE UTILITIES

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------------|--------------|----------|
| | | Initial | Extended |
| 1.1 SYSTEM GENERATION | | | |
| (a-c) The system must support the following types of generation: (a) complete system generation, (b) nucleus generation, (c) processor library generation. | 1 | | |
| (d-e) The system must support the following methods of performing system generation: (d) interactive macro instructions, (e) pre-defined control statements. | 2 | | |
| (f-i) The system must permit specification of: (f) memory partitions, (g) available resources, (h) number of interrupts, (i) interrupt hierarchy. | 3 4 4 | | |
| (j-k) The system must support specification of scheduling definition through: (j) priority levels, (k) scheduling algorithm. | 5 6 | | |
| (l-p) The system must support the following system specifications: (l) file specifications, (m) interrupt options, (n) accounting options, (o) error recovery procedures, (p) size of dynamic core allocation pool. | 3 7 8 | | |

1.1 SYSTEM GENERATION

Reference:

1. In a medium to large scale operating system, it is generally better to perform nucleus generation and processor library generation as separate phases rather than as a single step. However, smaller systems normally combine these two phases into a single complete system generation package.
2. The use of interactive as opposed to pre-defined control statements is a matter of personal preference. Either method is an acceptable way of performing system generation. The only environment where interactive techniques might prove more beneficial is in the "super-systems" such as CP-67 which permits several subordinate operating systems to time-share the computer. Since each user is then responsible for generating the operating system he will use, the interactive approach might prove handier.
3. The greater the number of specification options, the better a general purpose system can be tailored to a particular installation's requirements.
4. This criterion is highly desirable for a real-time processing system.
5. This technique is normally used for real-time processing systems.
6. This technique is normally used for batch processing systems.
7. If the system is to support applications that perform file manipulation, then the specification at system generation of such items as file sizes, privacy codes, maximum file tracks, maximum file counts, etc. are important.
8. In a facility that is supporting different environments the method of error recovery used may vary depending upon the environment in which the error occurred.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | | Reference | REQUIREMENTS | |
|--|--|-----------|--------------|----------|
| | | | Initial | Extended |
| 1.1 SYSTEM GENERATION (cont'd.) | | | | |
| (q-s) | The system must recognize the following specifications relative to users: <ul style="list-style-type: none"> (q) authorized users, (r) priority designations, (s) privacy codes. | | | |
| (t) | The system must have the capability to incorporate user-developed routines. | 9 | | |
| (u-x) | The system must provide the following support systems: <ul style="list-style-type: none"> (u) compilers, (v) utility programs, (w) data management systems, (x) assemblers. | 9 | | |
| (y) | The system must provide the capability to specify default options. | 9 | | |

1.1 SYSTEM GENERATION (cont'd.)

Reference:

9. This feature is highly desirable for most batch processing and time-sharing installations.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2 SYSTEM MAINTENANCE | | | |
| (a) The system must provide dynamic maintenance. | 1 | | |
| (b) The system must provide patching capability for system software. | 2 | | |
| (c-e) The system must support the following methods of system patching: (c) on-line, (d) temporary, (e) permanent. | 2 | | |
| (f-i) The system must permit user-controlled maintenance through: (f) new command definition, (g) command renaming, (h) operand renaming, (i) default value alteration. | 3 | | |

1.2 SYSTEM MAINTENANCE

Reference

1. This criterion is highly desirable for a real-time processing system. This feature allows the system to take the necessary steps to meet a changing hardware configuration or to operate in a degraded or fall-back mode with little if any external interaction.
2. This criterion should be specified for all processing systems. However, the installation should maintain stringent control over the use of the feature and restrict its use to system maintenance personnel.
3. This feature occurs in time-sharing processing systems and provides the user with limited system tailoring capabilities. The user can adapt certain system features to best meet his requirements without being detrimental to the use of the same features by other users.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | | Reference | REQUIREMENTS | |
|--|---|-----------|--------------|----------|
| | | | Initial | Extended |
| 2.1 | LIBRARY AND DIRECTORY MAINTENANCE | 1 | | |
| (a-b) | The system must provide dynamic cataloging support for: (a) load modules, (b) task/procedure definitions. | | | |
| (c-g) | The system must provide static cataloging support for: (c) load modules, (d) relocatable modules, (e) source modules, (f) macro routines, (g) task and job procedures. | | | |
| (h-k) | The system must provide the following utility functions: (h) copying libraries or library elements, (i) renaming library elements, (j) allocating/deallocating/repacking library space, (k) punching/listing/displaying library elements. | 2 | | |

2.1 LIBRARY AND DIRECTORY MAINTENANCE

Reference:

1. A system may provide both static and dynamic cataloging capabilities. Static cataloging requires the explicit execution of a library maintenance program to perform the cataloging function. Dynamic cataloging permits the user to add an element to a library without explicitly invoking a librarian program.
2. These criteria should be considered for specification for all systems supported by libraries.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.2 LOAD MODULE GENERATION | 1 | | |
| (a-b) The system must provide facilities for: | 2 | | |
| (a) program binding, | | | |
| (b) linkage resolution. | | | |
| (c) The system must support dynamic binding. | 3 | | |
| (d) The system must support static binding. | 3 | | |
| (e) The system must provide binder code-altering facilities. | 4 | | |
| (f-g) The system must resolve the following types of linkages: | 2,5 | | |
| (f) program control instructions, | | | |
| (g) data field references. | | | |
| (h) The system must provide an automatic system library scan for unresolved references. | 6 | | |
| (i) The system must provide a system library scan for unresolved references as a user option. | 6 | | |

2.2 LOAD MODULE GENERATION

Reference:

1. In most systems, capabilities exist to combine (bind) the object modules produced by various language compilers with subroutines and other object modules to produce a single program.
2. This criterion should be specified for all processing systems.
3. In dynamic binding the system performs the binding function at the command of an executing task; in static binding, binding is performed as an "off-line" step during program development.
4. This criterion is frequently desirable. It permits patches or altered code to be inserted in an object module during the binding process. While this could be done by reassembly or recompilation, it will be more economical to make small corrections at the binding stage.
5. It is quite likely that there will be a certain amount of interaction or branching between different modules of a linked job. In order for the programs to operate properly these areas of control must be defined and resolved during load module generation.

If any of the modules to be linked refer to data fields within, or created by other modules, or within existing library files, then the linkage program should also resolve these references.

6. During the linkage phase of program module generation, certain references may be made to items that are external to the system and reside in a system library. Unless all library elements are specifically included, some references to system library routines may be unresolved. Several systems provide a capability to scan the system library for the elements referenced. If found it is included within the composite load module. The library scan may be either automatic or a user option.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|----------------------------|--------------|----------|
| | | Initial | Extended |
| <p>4.1 PERIPHERAL DEVICE SUPPORT</p> <p>(a) The system must provide volume preparation facilities.</p> <p>(b) The system must provide volume maintenance facilities.</p> <p>(c-f) The system must provide support for the following types of volumes:</p> <p style="padding-left: 40px;">(c) tape,</p> <p style="padding-left: 40px;">(d) removable disks,</p> <p style="padding-left: 40px;">(e) removable drums,</p> <p style="padding-left: 40px;">(f) data cells.</p> | <p>1</p> <p>2</p> <p>3</p> | | |
| <p>4.2 SYSTEM SIMULATION ROUTINES</p> <p>(a-b) The system must provide the following types of simulation routines:</p> <p style="padding-left: 40px;">(a) system facilities,</p> <p style="padding-left: 40px;">(b) communication facilities.</p> | <p>1</p> | | |

4.1 PERIPHERAL DEVICE SUPPORT

Reference:

1. This function is normally invoked when a new volume is to be added as a system component. The volume preparation function writes a standard system label on the volume, formats the records and/or tracks where track formatting is required (such as for disk files), creates any necessary volume directory entries, and allocates space for additional directories, communication buffers, etc. Only upon completion of this function is a volume available for normal system use.
2. Volume maintenance functions may also provide diagnostic routines to verify the correctness of all volumes in the system. Normally, these routines perform a surface analysis of the volume to detect failing surface areas and identify or replace any defective areas. Certain file purging functions may also be invoked to erase selected areas on a volume or to clear main or secondary storage.
3. This criterion is entirely dependent upon the hardware to be used by the system and can only be specified when the hardware is known.

4.2 SYSTEM SIMULATION ROUTINES

Reference:

1. These criteria should be considered for specification for most real-time processing systems as well as systems that will foster large development efforts. The system facilities to be simulated would be such items as I/C activity, interrupts, etc., while the communication facilities simulated would consist of such items as message transmission, receipt, etc.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-------------------|--------------|----------|
| | | Initial | Extended |
| <p>4.3 SYSTEM MEASUREMENT ROUTINES</p> <p>(a-b) The system must support the following measurement concepts:</p> <p>(a) sampling, (b) intercept.</p> <p>(c-e) The system must support the following methods of measurement:</p> <p>(c) software, (d) hardware, (e) mixed.</p> | <p>1</p> <p>2</p> | | |
| <p>4.4 STAND-ALONE UTILITIES</p> <p>(a-b) The system must support the following types of stand-alone utilities:</p> <p>(a) status displays, (b) recovery support.</p> | <p>1</p> <p>2</p> | | |

4.3 SYSTEM MEASUREMENT ROUTINES

Reference:

1. This function is highly desirable for most processing systems and greatly aids an installation in tailoring the system to best meet operational requirements. While the sampling concept of measurement produces utilization factors at periodic intervals, the intercept method provides a view of total utilization by recording all functions of interest. The intercept method provides a more comprehensive view of total system activity; however, it is somewhat more time consuming.
2. Very few systems provide adequate methods of measuring system performance. Methods presently used consist of software statistics routines, measurement devices built into the system hardware, and mixtures of both methods.

4.4 STAND-ALONE UTILITIES

Reference:

1. This criterion should be considered as an aid in performing diagnostic maintenance.
2. This criterion is highly desirable for a real-time processing system and is quite useful for most other processing systems.

3.3.3 Third Level of Detail (Part II - System Management Functions)

This subsection covers the following third level system management functions:

- 4.1.1 VOLUME PREPARATION
- 4.1.2 VOLUME MAINTENANCE
- 4.2.1 SIMULATION OF SYSTEM FACILITIES
- 4.2.2 SIMULATION OF COMMUNICATION FACILITIES
- 4.4.1 STAND-ALONE STATUS DISPLAY
- 4.4.2 STAND-ALONE RECOVERY SUPPORT

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|--|--------------|----------|
| | | Initial | Extended |
| <p>4.1.1 VOLUME PREPARATION</p> <p>(a-e) The system must support the following record formats:</p> <ul style="list-style-type: none"> (a) blocked/unblocked, (b) hardware keys, (c) software keys, (d) variable length, (e) fixed length. <p>(f-g) The system must support the following directory formats:</p> <ul style="list-style-type: none"> (f) system specified, (g) user specified. <p>(h) The system must provide for directory levels.</p> <p>(i-k) The system must support the following types of space allocation:</p> <ul style="list-style-type: none"> (i) file space, (j) dynamic communication buffer areas, (k) workspace. <p>(l-m) The system must provide the following space allocation modes:</p> <ul style="list-style-type: none"> (l) record, (m) block, (n) track, (o) cylinder, (p) segment. | <p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> | | |

4.1.1 VOLUME PREPARATION

Reference:

1. Blocked/unblocked record formats are used in processing data for transmission between storage devices. The blocking of records is used to improve data rates and conserve storage space on the storage device by reducing the number of interrecord gaps in the file.

Hardware and software keys are usually associated with direct access devices as a form of record labeling. The hardware key is usually a fixed designator that never changes while the software key is normally a data item within the record.

Since the types of records handled by a system can vary due to the data or applications programs, the capability should be provided to support both variable and fixed length record formats.

2. Directories are usually provided by a system to delineate the files, programs, etc., available or active within the system. The use of system specified directory formats is the standard method of directory organization. The support by a system of user created directory formats should only be considered in special development activities.
3. If the system is to support the creation of files, then it should support the allocation of areas for building or creating files on storage devices.
4. Systems that support the processing of communication information should support the allocation of communication buffer areas, since the system frequently may not be able to process this data as it occurs but must hold it for processing at a later time.
5. It is usually desirable to provide several levels of space allocation. Large allocation blocks are desirable for creating file and workspace areas. However, the system should also provide a lower level of allocation equal to the level to which the data element can be individually addressed.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 4.1.2 VOLUME MAINTENANCE | | | |
| (a-b) The system must provide surface analysis for the following devices: (a) magnetic tape, (b) direct access. | 1,2 | | |
| (c-d) The system must provide for track replacement: (c) automatically, (d) by operator command. | 3 | | |
| (e-f) The system must provide the following types of purge routines: (e) erase tape, (f) overwrite direct access. | 1 | | |
| (g) The system must provide purge verification. | 4 | | |

4.1.2 VOLUME MAINTENANCE

Reference:

1. These criteria should be specified for all systems whose hardware configuration contains these device types.
2. Surface analysis is a diagnostic function which determines whether a device surface area is performing properly. This type of verification should be available for all recording or data holding devices.
3. These criteria should be considered for specification for all systems with the automatic mode being highly desirable for a real-time processing system. During operation with a storage track, an error may occur which indicates a defective track. The automatic switch to an alternate track by the system allows continuous execution without external interaction. If the system does not provide automatic switching to an alternate track, the operator should be allowed to establish an alternate track and initialize the program at a point where the first output to the defective track was performed.
4. This criterion should be specified for all processing systems that perform purging functions. Usually when a system supports the purging of storage elements, the purging pattern is read and checked for verification. A diagnostic check of storage areas is frequently a by-product of this function.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-------------------|--------------|----------|
| | | Initial | Extended |
| <p>4.2.1 SIMULATION OF SYSTEM FACILITIES</p> <p>(a-d) The system must provide simulation of the following system facilities:</p> <p>(a) I/O device activity, (b) real-time interrupts, (c) error indicators, (d) executive system functions.</p> | <p>1</p> <p>2</p> | | |
| <p>4.2.2 SIMULATION OF COMMUNICATION FACILITIES</p> <p>(a-c) The system must provide simulation of the following communication facilities:</p> <p>(a) message transmission, (b) message receipt, (c) exception processing.</p> | <p>1</p> | | |

4.2.1 SIMULATION OF SYSTEM FACILITIES

Reference:

1. The simulation of I/O device activity and real-time interrupts is important during program checkout and development. Simulation of error indications is also extremely useful to debug diagnostic software routines. If the system is to support real-time processing, or checkout is to be performed during support of on-line operations, or if the executive system is under development, then simulation of system facilities should be considered.
2. The simulation of executive system functions is highly desirable during the development of a large comprehensive operating or real-time system. By simulating these functions, support programs may be tested and debugged without requiring a completed version of the operational executive system. Thus, executive system development may parallel, rather than precede, support program development.

4.2.2 SIMULATION OF COMMUNICATION FACILITIES

Reference:

1. These criteria should be considered for specification if the system is to support program development for teleprocessing or any mode of interactive communication.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| <p>4.4.1 STAND-ALONE STATUS DISPLAY</p> <p>(a-e) The system must provide stand-alone status displays of the following:</p> <ul style="list-style-type: none"> (a) core storage, (b) hardware registers, (c) file storage, (d) logout areas, (e) read-only storage. | 1 | | |
| <p>4.4.2 STAND-ALONE RECOVERY SUPPORT</p> <p>(a-e) The system must provide the following types of stand-alone recovery support:</p> <ul style="list-style-type: none"> (a) rebuild message queues, (b) rebuild system processing queues, (c) reconstruct on-line file transactions, (d) re-initiate suspended processing, (e) re-establish communication line links. | 1 | | |

4.4.1 STAND-ALONE STATUS DISPLAY

Reference:

1. The capability to obtain a stand-alone status display of each of these areas is highly desirable for maintenance after a system failure.

4.4.2 STAND-ALONE RECOVERY SUPPORT

Reference:

1. Since stand alone routines for recovery are only invoked when all other automatic attempts for recovery have been exhausted, the capability presented here is a last-chance type of control. Consequently, the critical elements for continued processing should be emphasized, while the aesthetic features should be ignored. Thus, if the system is message based, then message queues should be rebuilt if there is no way to request the terminals to retransmit incompletd messages. Similarly, if communication line links are not easily restored, then a routine to perform this should be provided. A word of caution, however; these are individual routines executed independently and as a last resort; consequently, the system will remain inoperational until all selected routines have been executed. If mean time to recovery is a critical factor, these routines should be short and few.

3.4 Requirements Checklist - Part III - Data Manipulation Functions

The following subsections present specification and evaluation criteria for the components of the operating system that permit the user to access and process data.

3.4.1 First Level of Detail (Part III - Data Manipulation Functions)

This subsection covers the following first level data manipulation functions:

- 1.0 DATA MANAGEMENT
- 2.0 DATA HANDLING UTILITIES
- 3.0 SORTING AND MERGING

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| <p>1.0 DATA MANAGEMENT</p> <p>(a) The system must provide file management facilities.</p> <p>(b) The system must provide I/O support facilities.</p> <p>(c) The system must provide an independent data management system.</p> | 1 | | |

1.0 DATA MANAGEMENT

Reference:

1. The data management functions consist of file management facilities, I/O support facilities, and data management system facilities. File management facilities provide file support for the system files as entities rather than for the individual data records within the files. Support for the latter is provided by the other two data management categories.

I/O support facilities enable a program to access and process individual data records within the file. These facilities are normally invoked by issuing macro instructions within the problem program and eliminate the need for programmers to be concerned with many of the problems of reading and writing data records.

Data management systems allow a user with limited programming interests to perform certain functions on a data file, such as retrieving and displaying selected portions of the file. Generally, data management systems are adjuncts to an operating system and are more or less self-contained depending upon the architecture of the particular system. Consequently, a data management system will make use of many of the features provided by other operating system functions in its own internal design.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.0 DATA HANDLING UTILITIES | 1 | | |
| (a) The system must provide facilities for display support. | 2 | | |
| (b) The system must provide facilities for peripheral device support. | 2 | | |
| (c-d) The system must provide the following types of utilities: | 3 | | |
| (c) generated code, | | | |
| (d) interpretive code. | | | |

2.0 DATA HANDLING UTILITIES

Reference:

1. The data handling utility programs are generally rather uncomplicated routines of general usefulness to the installation. Classically, these routines were independent programs which were loaded and executed when required. However, in contemporary systems they have been incorporated into the operating system and are activated by a program call, a system control card, or an operator key-in. These utilities rarely interface directly with an executing program, though they are frequently included as a separate job step in a multi-step job.
2. The utility routines that produce visual output as final products are termed display facilities. While the printer is the most common output medium, CRT's, console typewriters, etc., may also be utilized by various display routines.

The utility routines that provide peripheral device support perform such functions as volume positioning, media conversion, data editing, and test file support.

These criteria should be specified for all batch processing and time-sharing systems.

3. While frequently employed for evaluation, this criterion is rarely specified. In general, generated code routines are somewhat more complex and faster in execution than interpretive code.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 3.0 SORTING AND MERGING | 1 | | |
| (a) The system must provide a balanced merge technique. | 2,3 | | |
| (b) The system must provide an oscillating sort technique. | 2,4 | | |
| (c) The system must provide a polyphase sort technique. | 2,5 | | |
| (d) The system must provide a cascade sort technique. | 2,6 | | |
| (e) The system must permit the use of sequential devices for external workspace. | 7 | | |
| (f) The system must permit the use of random access devices for external workspace. | 7 | | |

3.0 SORTING AND MERGING

Reference:

1. Programs that sort or merge sets of data records normally comprise a part of the operating system. The sorting function takes strings of unordered records and sequences them according to a given key or set of keys within each record. The merging function, on the other hand, takes separate strings of records which have already been ordered by keys and produces a composite ordered output string. The design of most sort programs is such that the unordered records are placed into ordered strings and then merged. Consequently, a single program usually suffices to perform both functions.
2. While frequently employed for evaluation, this criterion is rarely specified, however, it may be possible to specify a sorting technique based on previous experience or based on the hardware configuration supported by the system.
3. A balanced two-way merge requires four tape units, two for output and two for input. In a multiway, balanced merge any number of tapes can be used for input as long as the same number are available for output. In other words, if there are "N" input tapes, then the tapes required for a multiway balanced merge must be "2N."
4. An oscillating sort is one in which there is an oscillation between the internal sort (string creation) and the merge phase. Internal sort creates an ordered string of records on all but one of the secondary storage work files. The merge phase then sequences these strings from each work area onto the remaining work file. Control is then returned to the internal sort phase to create new input strings on the N-1 work files. Thus control oscillates between string creation and string merging.
5. Polyphase sort makes use of all available tape drives to perform a very high order of merge. The order of merge is "N-1" where N is the number of tapes available. The input is distributed onto the N-1 tapes. Then each of these units is read as the merged records are written on the "Nth" unit. Polyphase sorting is normally only advantageous for magnetic tapes that have a hardware "read backward" capability.
6. Like polyphase sorting, cascade sorting uses fewer tapes to do higher order merges. For example, with five tapes, one as input, items are distributed unequally onto four drives. All four tapes are then merged (four-way merge) onto the input tape and since one of the tapes has few strings it becomes empty first. The merge then continues as a three-way merge, two-way merge, and finally a copy operation.
7. This criterion is highly dependent upon the hardware configuration and can only be specified when the hardware configuration is known.

3.4.2 Second Level of Detail (Part III - Data Manipulation Functions)

This subsection covers the following second level data manipulation functions:

- 1.1 FILE MANAGEMENT FACILITIES
- 1.2 I/O SUPPORT FACILITIES
- 1.3 DATA MANAGEMENT SYSTEM FACILITIES
- 2.1 DISPLAY FACILITIES
- 2.2 PERIPHERAL DEVICE SUPPORT
- 3.1 SORT MODULE DEVELOPMENT
- 3.2 SORT MODULE EXECUTION

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-------------------------------------|--------------|----------|
| | | Initial | Extended |
| <p>1.1 FILE MANAGEMENT FACILITIES</p> <p>(a-c) The system must provide the following file management capabilities:</p> <p>(a) file location recognition,</p> <p>(b) file security controls,</p> <p>(c) backup and restoration facilities.</p> | <p>1</p> <p>2</p> <p>3</p> <p>2</p> | | |
| <p>1.2 I/O SUPPORT FACILITIES</p> <p>(a-b) The system must provide the following types of I/O support facilities:</p> <p>(a) physical I/O,</p> <p>(b) logical I/O.</p> | <p>1</p> | | |

1.1 FILE MANAGEMENT FACILITIES

Reference:

1. File management functions are invoked to locate on-line and off-line files, permit or restrict user access to files, and to provide backup and restoration services in case of file damage.
2. This criterion should be specified for all processing systems.
3. This criterion should be specified for all time-sharing and most batch processing systems supporting diverse users.

1.2 I/O SUPPORT FACILITIES

Reference:

1. An important distinction is made in describing the levels of I/O support provided by a system. The basic level of support, physical input/output, is provided by routines that initiate the actual data transmission process and provide program access to the data in the format of transmission.

The extended level of support, logical input/output, allows manipulation of data without regard to the physical structure of the data. It thus serves as an intermediary between user data handling operations and the physical input/output services of the system.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3 DATA MANAGEMENT SYSTEM FACILITIES | 1 | | |
| (a-d) The system must provide data management facilities with the following capabilities: (a) data base creation, (b) data base modification, (c) data base interrogation, (d) report development. | 2 | | |
| (e-f) The system must provide data management facilities that operate in the following modes: (e) batch, (f) interactive. | 3 | | |

1.3 DATA MANAGEMENT SYSTEM FACILITIES

Reference:

1. A data management system is a group of integrated routines developed to create and maintain an organized collection of related data, known as the data base, and to interrogate the data base and produce various types of formatted reports. A data management system will create files from various input sources; maintain these files by additions, deletions, and alterations; create new files and reorganize and merge existing files; select data via user-prepared queries; make computations on this data; and produce reports in system-defined or user-specified formats. A data management system may be designed to operate in either a batch or interactive mode.
2. Normally, a data management system consists of each of these elements. However, there may be unusual circumstances whereby one or more of these elements will not be required. For example, if the installation intends to develop its own report generation capability, this feature need not be included within the DMS.
3. This criterion is directly derived from the intended operational environment.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 2.1 DISPLAY FACILITIES | 1 | | |
| (a) The system must allow formatted display. | 2 | | |
| (b) The system must allow unformatted display. | 3 | | |
| (c-e) The system must provide utilities for the following types of display devices: | 4 | | |
| (c) printer, | | | |
| (d) typewriter, | | | |
| (e) CRT. | | | |
| 2.2 PERIPHERAL DEVICE SUPPORT | | | |
| (a-d) The system must provide the following peripheral device support facilities: | | | |
| (a) volume positioning, | 1 | | |
| (b) media copying, | 2 | | |
| (c) data editing, | | | |
| (d) text data file support. | 3 | | |

2.1 DISPLAY FACILITIES

Reference:

1. The most common display facilities provided are for main storage (core dumps), system catalogs, tables and directories. Other display facilities are generally provided for data stored on any secondary storage media supported by the system.
2. This mode of display is advantageous when the recipient is concerned with the contents of the data being displayed, rather than its machine structure.
3. This function is very important if a picture image of the exact data structure is required.
4. The printer is the most frequent means of displaying data. However, a remote user may only have a typewriter or CRT available.

2.2 PERIPHERAL DEVICE SUPPORT

Reference:

1. This function consists of such features as rewinding, backspacing, or forward spacing a magnetic tape, moving a disk arm, etc.
2. Media copying facilities should normally be available to all facility users. They consist of routines to accomplish such functions as tape to disk, tape to card, tape to tape, card to tape, etc.
3. This is a very useful feature to a facility and should be specified if file-oriented program development is to be a major installation objective.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 3.1 SORT MODULE DEVELOPMENT | | | |
| (a) The system must provide a tailorable general purpose sort program. | 1 | | |
| (b-c) The system must provide parameter specification by: | 2 | | |
| (b) control cards, | | | |
| (c) internal linkage parameters. | | | |
| (d) The system must determine the allocation of internal workspace. | 3 | | |
| (e) The system must allow a supplied parameter to determine internal workspace allocation. | 3 | | |
| (f) The system must determine external workspace allocation. | 3 | | |
| (g) The system must allow a supplied parameter to determine external workspace allocation. | 3 | | |
| (h-j) The system must provide the following options: | 4 | | |
| (h) ascending/descending output sequence, | | | |
| (i) single/multiple sort control fields, | | | |
| (j) single/mixed data field formats. | | | |
| (k-n) The system must recognize the following field keys. | 5 | | |
| (k) alphanumeric, | | | |
| (l) binary, | | | |
| (m) zoned/packed decimal, | | | |
| (n) floating point. | | | |
| (c) The system must support a user-specified collating sequence. | 6 | | |

3.1 SORT MODULE DEVELOPMENT

Reference:

1. This technique is prevalent in most sort packages and allows the user to tailor the sort program either through control statements or selectable subroutines. In this way the user can create the most efficient program for his specific need.
2. Many systems provide for parameter specification either through control cards or internal macro instructions. The use of control cards is recommended for "off-line" sorting of fixed data files. The use of internal macros for specifying parameters is most desirable when the sort program operates as an in-line routine for a more comprehensive application package.
3. It is generally advisable to have the system perform all workspace calculations. However, there may be overriding reasons for allowing a user to specify these as parameters - particularly if the generated sort routine is to be a small part of a more comprehensive application program package.
4. Most systems provide the capability to specify ascending, descending, or a mixture of an ascending/descending output sequence. This allows the user to specify different sort orders based upon different keys.

Single sort control fields require that the sort key consist of an adjacent set of characters. Multiple control keys permit sorting using a number of data elements that need not be contiguous.

Some systems do not provide the capability for sorting mixed files containing different kinds of records or merging files with fixed and variable record length. While other systems provide these features as options to increase user flexibility and to lessen the requirement for restructuring files prior to sort.

5. It is generally advisable to have the sort program recognize every type of data element representation permitted within the system.
6. This function rarely occurs in standard system supplied sort programs. However, it is sometimes necessary for a user to produce sequences based upon an alternate collating sequence. For example, if the data is sorted on one system but intended for a different system (with a different collating sequence), it would be worthwhile to use the second system's collating sequence.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-------------------------------------|--------------|----------|
| | | Initial | Extended |
| <p>3.2 SORT MODULE EXECUTION</p> <p>(a-c) The system must provide the following types of sorting:</p> <ul style="list-style-type: none"> (a) full data records (record sort), (b) sort key and record address (tag sort), (c) sort key and selected fields (field select sort). <p>(d) The system must provide control for workspace overflow.</p> <p>(e-i) The system must permit the inclusion of user codes relative to:</p> <ul style="list-style-type: none"> (e) label processing, (f) input record insertion/modification/deletion, (g) output record insertion/modification/deletion, (h) blocking/deblocking control, (i) error processing <p>(j) The system must provide an automatic final pass sequence check.</p> <p>(k) The system must provide an optional final pass sequence check.</p> | <p>1</p> <p>2</p> <p>3</p> <p>3</p> | | |

3.2 SORT MODULE EXECUTION

Reference:

1. Three types of sorting are most generally used: a record sort, where the full record is sorted; a field select sort, where certain fields are deleted from the record prior to sorting; and a tag sort, where the full record is stored on an intermediate device and only its sort key and address are sorted. The latter is primarily of advantage when the sort program is used as a supporting task of another executing application program.
2. Exits to user codes within a sort module provides the user with the flexibility to perform any of the listed criteria without modifying the standard sort module. Each user can then perform any of the unique functions required without affecting the more general sort module structure.
3. It is usually a good practice to perform a final pass sequence check to validate the results of the sort/merge. Whether or not this is performed automatically or as an option depends upon the operational philosophy of the installation.

3.4.3 Third Level of Detail (Part III - Data Manipulation Functions)

This subsection covers the following third level data manipulation functions:

- 1.1.1 FILE LOCATION RECOGNITION
- 1.1.2 FILE ACCESS CONTROL
- 1.1.3 BACKUP AND RESTORATION CAPABILITIES
- 1.2.1 DATA ACCESS CONTROL
- 1.2.2 DATA BLOCKING/DEBLOCKING CONTROL
- 1.2.3 LABEL PROCESSING
- 1.3.1 CONTROL SPECIFICATION
- 1.3.2 DATA FILE GENERATION AND MAINTENANCE
- 1.3.3 DATA QUALIFICATION AND RETRIEVAL
- 1.3.4 DATA OUTPUT
- 2.1.1 UNFORMATTED DISPLAY
- 2.1.2 FORMATTED DISPLAY
- 2.2.1 VOLUME POSITIONING
- 2.2.2 MEDIA COPY FACILITIES
- 2.2.3 DATA EDITING FACILITIES
- 2.2.4 TEST DATA FILE SUPPORT

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.1 FILE LOCATION RECOGNITION | 1 | | |
| (a-b) The system must provide for locating files using the following methods: | 2 | | |
| (a) cataloged addresses, | 2 | | |
| (b) label recognition. | | | |
| (c-d) The system must use a file identifier which is: | 3 | | |
| (c) system assigned, | | | |
| (d) user assigned. | | | |
| (e) The system must support multiple cataloging levels. | 4,2 | | |
| (f) The system must maintain separate catalogs. | 5,2 | | |

1.1.1 FILE LOCATION RECOGNITION

Reference:

1. In most systems permanent data files are identified by labels assigned by the user or the system. File labels may be composed of such items as the file identifier, the file edition number, the owner's name and account number, and perhaps a file utilization privacy code.
2. Maintaining a catalog of file locations is the most expeditious method used to locate files in that the system merely searches the catalog for the address of the referenced file.
3. Both methods of label assignment are presently used. With system assigned labels, the system is responsible for the uniqueness of the labels; with user assigned labels the user must assume the responsibility for creating a unique label. Many systems provide for mixed label creation in which the user identifier is used by the system in conjunction with a system generated label.
4. Some systems provide multiple levels of cataloging. Though not extensively used, this feature may be of advantage when a hierarchical structure of data base management is desired. For example, a payroll file could be constructed to consist of three subfiles: administrative staff payroll, professional staff payroll, and operational staff payroll. The professional staff payroll file could also be made into multiple files: managerial staff, corporation officers, and consultants. Thus, the number of records composing a file is a function of the file cataloging level.
5. Certain systems provide for an active or master catalog and a temporary catalog which is used for programs that are in a checkout phase and which would not be placed on the master catalog until operational.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.1.2 FILE ACCESS CONTROL | 1 | | |
| (a) The system must provide file security control. | 2 | | |
| (b) The system must provide read/write access control. | | | |
| (c) The system must provide concurrent access control. | 3 | | |
| (d-h) The system must provide file access protection for the following units: | 4 | | |
| (d) volumes, | | | |
| (e) files, | | | |
| (f) physical records, | | | |
| (g) logical records, | | | |
| (h) data elements. | | | |
| (i-j) The system must provide for access control maintenance: | 5 | | |
| (i) by system supervisor, | 2 | | |
| (j) by programming conventions. | | | |

1.1.2 FILE ACCESS CONTROL

Reference:

1. The designation and restriction of file access may be a function under control of the operating system or it may merely be established by a set of installation programming conventions. When controlled by the system, the owner of a file can usually designate that the file is to be maintained for his use only, for the use of a designated group only, or for general use. Frequently, the owner may also specify the level of access afforded each designated user. For example, access may be restricted to a read-only level for some users while others are allowed full read and write capabilities.
2. This criterion should be specified for all processing systems that maintain classified or private information.
3. Concurrent access control is required to maintain the scheduling and handling of concurrent or simultaneous requests for a data file from separate programs or users in a multiprogramming or time-sharing environment. Basically, the control function must determine if multiple user access can be permitted or whether the file must be restricted to single user access. In situations where multiple users may simultaneously access a single file, it is usually desirable to grant any number of them read-only access, but to restrict write access to a single user at a time.
4. The level of file access protection is a part of the overall philosophy of data base design. One school of thought advocates a single large data base with individual records and data elements individually allocated or denied to certain users. Another school advocates a data base consisting of multiple files with the user either allowed or denied access to the entire file.
5. Either of these criteria can be specified as a method of maintaining access control. However, system supervisory control is considerably more reliable; the other method is dependent upon external users adhering to the conventions. System supervisory control should be specified for time-sharing and multiple-user batch processing systems.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-------------------------------------|--------------|----------|
| | | Initial | Extended |
| <p>1.1.3 BACKUP AND RESTORATION CAPABILITIES</p> <p>(a-b) The system must support the following backup media:</p> <p>(a) on-line devices, (b) off-line devices.</p> <p>(c-d) The system must provide the following restoration techniques:</p> <p>(c) automatic, (d) operator initiated.</p> <p>(e-g) The system must provide the following file backup techniques:</p> <p>(e) grandfather files, (f) periodic checkpoints, (g) retention of transaction data.</p> | <p>1</p> <p>2</p> <p>2</p> <p>3</p> | | |

1.1.3 BACKUP AND RESTORATION CAPABILITIES

Reference:

1. This function is highly dependent upon the hardware configuration and can only be specified when the hardware configuration is known. Syst support of on-line backup devices can significantly improve the system recovery time and should be considered for a real-time processing syste
2. This criterion is highly desirable for a real-time processing system.
3. The use of grandfather files as a file backup technique provides a secur base from which a system can be restored to operation. However, this type of file backup is usually associated with a periodic update of the grandfather file which means that file changes made during operation of the system must be retained and re-run to bring the grandfather file up to date. This procedure is normally used for batch processing commerca lly oriented jobs.

Periodic checkpoints are used as a backup technique in many real-time processing environments. However, the file maintained on the check point will not be totally up to date and intermediate transaction data should be retained.

The retention of all transaction data is an excellent technique to be used in conjunction with redundant files or grandfather files. In this way the redundant file or grandfather file can be updated to the status of the previous working file.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.1 DATA ACCESS CONTROL | | | |
| (a-e) The system must support the following access methods: | 1 | | |
| (a) sequential, | 2 | | |
| (b) random (direct), | 3 | | |
| (c) keyed, | 4 | | |
| (d) indexed, | 5 | | |
| (e) teleprocessing. | 6 | | |
| (f-i) The system must permit data access at the following levels: | 7 | | |
| (f) block, | | | |
| (g) record, | | | |
| (h) data item, | | | |
| (i) character. | | | |

1.2.1 DATA ACCESS CONTROL

Reference:

1. The specification of these criteria is highly dependent upon the system hardware configuration.
2. Sequential access methods process records serially and read or write them consecutively on a storage volume. Sequential access may be provided for data stored on any secondary storage medium; although, certain storage media, such as magnetic tape, obviously dictate a file organization of this type.
3. Random access methods read and write records without regard to the physical sequence in which they are stored. Consequently, records stored in this type of organization must have some type of identification code that will enable the record location to be determined. Usually, an algorithm is used to convert a unique record identification into a unique storage address on a random access storage device.
4. Keyed access methods rely on a selected data field within each record to uniquely identify the record. This data field, called the record key, frequently corresponds to the record identification code, though it need not do so. Keyed access is useful for secondary storage devices which have a physical design that features hardware-implemented search instructions. In such systems, a record request will cause the read/write mechanism of the storage medium to scan the entire file in search of a selected record key. The technique is advantageous since processor time need not be spent in scanning secondary storage and may thus be employed with other execution functions. The technique is also frequently augmented by software which isolates a portion of the file prior to issuing the keyed search in order to avoid a scan of the entire file.
5. Indexed access methods create and maintain files which, in addition to the data record, have directories of selected record field values and their corresponding record addresses. Records may then be located by searching the directory rather than the file. Some form of immediate access storage, such as disk, is necessary for indexed access methods to have value.
6. Teleprocessing data is usually composed of character strings of varying lengths. While not a file in the classical definition, most systems provide assistance in accessing and processing the relatively unformatted messages that accumulate in the system communication buffers. This assistance normally handles all communication between the computing system and remotely located terminals.
7. The level to which access is provided greatly increases user capability with a corresponding increase in system overhead.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.2 DATA BLOCKING/DEBLOCKING CONTROL | 1 | | |
| (a) The system must provide record acquisition by deblocking. | | | |
| (b) The system must provide move mode deblocking. | 2 | | |
| (c) The system must provide locate mode deblocking. | 2 | | |
| (d-f) Blocking/deblocking facilities must be available for: | 3 | | |
| (d) fixed length, | | | |
| (e) variable length, | | | |
| (f) records of undefined length. | | | |
| (g) The system must provide system specified block sizes. | | | |
| (h) The system must permit user-specified block sizes. | | | |

1.2.2 DATA BLOCKING/DEBLOCKING CONTROL

Reference:

1. Blocking combines two or more individual records into one physical data block; deblocking isolates the individual records within a physical data block. Record lengths may be fixed, variable or undefined and all of these types may be blocked or unblocked.
2. While frequently employed for evaluation these criteria are rarely specified. The locate mode of deblocking provides the user with the capability to locate and process a specific record through the use of pointers without physically moving the record to a processing area.

In move mode deblocking the system physically moves each deblocked record from the I/O buffer area into a user storage area. The locate mode is thus somewhat faster and requires less core storage. It may, however, also introduce unnecessary complexity in record processing for the application programmer.
3. This function is usually supported by all processing systems that provide logical input/output support.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.3 LABEL PROCESSING | 1 | | |
| (a) The system must provide an automatic label generation capability. | 2 | | |
| (b) The system must allow users to generate labels. | 3 | | |
| (c) The system must provide automatic label checking facilities. | 2 | | |
| (d) The system must allow label checking upon user request. | | | |

1.2.3 LABEL PROCESSING

Reference:

1. Most systems provide facilities for writing and checking file labels when a data file is opened or closed. Many systems also permit the user to specify his own labels and to supply special routines for processing them. A few of the smaller systems provide no label generation and checking facilities, and all label processing functions must be performed by the user.
2. This function is usually provided in medium to large processing systems and should be specified.
3. Many systems support user generated labels. User labels also require special user routines to read and validate the labels.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.1 CONTROL SPECIFICATION | 1 | | |
| (a-d) The system must allow the specification of the following types of formats: (a) file formats, (b) report formats, (c) input formats, (d) query formats. | 2 | | |
| (e-f) The system must provide the following methods of deriving control functions: (e) generative routines, (f) interpretative routines. | 3 | | |

1.3.1 CONTROL SPECIFICATION

Reference:

1. A data management system must be provided with a set of specifications or commands delineating the job it is to perform. The system interprets these specifications, and generates functional modules to perform the selected jobs. These modules may take the form of interpretive tables or executable programs or subroutines. Generally, the system will maintain an extensive library of functional subroutines which may be incorporated as needed into the final support modules. These subroutines range from arithmetic and data conversion routines to file searching and positioning capabilities. The generation of the resulting support modules is analogous to the process used by a compiler to convert user code into machine executable code.
2. These formats should be specified for each of the elements selected to compose the data management system (see 1.3 a,b,c,d).
3. The generative type of derivation is a process which actually structures a machine executable module to perform the required function.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|---|--------------|----------|
| | | Initial | Extended |
| <p>1.3.2 DATA FILE GENERATION AND MAINTENANCE</p> <p>(a-e) The system must provide the following data file generation and maintenance facilities:</p> <ul style="list-style-type: none"> (a) fixed input transaction processing, (b) logical (programmed) file maintenance, (c) interactive file maintenance, (d) file reorganization, (e) data error procedures. <p>(f-g) The system must allow the following types of correction:</p> <ul style="list-style-type: none"> (f) interactive, (g) pre-established. | <p>i</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>6</p> | | |

1.3.2 DATA FILE GENERATION AND MAINTENANCE

Reference:

1. Data file generation and maintenance is concerned with defining the internal structure of a file, allocating space for the file on a storage device, processing input transactions against the file, performing logical and interactive maintenance on the file, and reorganizing the file when the structure must be modified.
2. Fixed input transaction processing involves defining input data element formats, validating the data elements as they are entered, converting them into internal formats, and storing them in the data file. Input format definitions may be established prior to the entry of the data into the system or the data entry may be self-defining.
3. Logical file maintenance permits conditional or programmed updating of a data file. Logical maintenance may or may not be transaction oriented. If it is, the transaction updates the object file only when specified criteria are satisfied. Non-transaction oriented maintenance is usually accomplished via internal actions generated by a computer pseudo-language program. For example, a logical maintenance job might specify the deletion of every record written after a given calendar date, or conversely, the retention of only those records written between two given calendar dates.
4. Interactive maintenance, as its name implies, is the updating of a data file from an on-line terminal. Almost all interactive maintenance applications utilize logical file maintenance features. Prior to initiating the actual transaction the terminal user must usually establish logical parameters to isolate records of interest.
5. This function provides for reorganization of one or more existing data files into a new composite output file. Data fields may be added, deleted, or changed in size or type under the restructuring process.
6. Most data management systems provide for the use of pre-established actions to be performed in case of an error. A more unique capability is that of on-line interactive correction support whereby the interactive user can correct data errors as they are recognized.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.3 DATA QUALIFICATION AND RETRIEVAL | 1 | | |
| (a-b) The system must allow the following interrogation modes: | | | |
| (a) interactive, | | | |
| (b) batch. | | | |
| (c-e) The system must allow retrieval mode control through the following pre-stored queries: | | | |
| (c) fixed logic, | 2 | | |
| (d) modifiable logic, | 3 | | |
| (e) parameterized. | 4 | | |
| (f-h) The system must allow retrieval mode control through the following interactive queries: | | | |
| (f) cue-response format, | 5 | | |
| (g) prompting format, | 6 | | |
| (h) programmable format. | 7 | | |
| (i-m) The system must allow query processing through the following types of logical operators: | 8 | | |
| (i) Boolean, | 9 | | |
| (j) quantitative (occurrence), | 10 | | |
| (k) arithmetic, | 11 | | |
| (l) statistical, | 12 | | |
| (m) application-defined. | 13 | | |
| (n) The system must allow nested logical operators. | | | |
| (o-r) The system must allow the following query processing operand formats: | 8 | | |
| (o) constant value, | | | |
| (p) another data field, | | | |
| (q) interim result, | 14 | | |
| (r) arithmetic expressions. | | | |
| (s-u) The system must support the following types of file searches: | | | |
| (s) single file, | | | |
| (t) multi-file, | | | |
| (u) inter-file. | | | |

1.3.3 DATA QUALIFICATION AND RETRIEVAL

Reference:

1. Data management systems may be designed so that data qualification and retrieval can be accomplished in either a batch or interactive mode. In the batch mode the data base may be interrogated by individually prepared logical queries or by pre-stored logical queries in which specific operands can be altered. In the interactive mode interrogation may be by a cue-response form of query, by a prompting query which "guides" the user through the interrogation, or by a user-programmed query. Each record satisfying the parameter of the query may be directly displayed, retained on an intermediate file for subsequent processing, or passed on to another portion of the data management system, such as data output or file maintenance. Thus, a single data qualification scheme generally serves the entire data management system.
2. Fixed logic pre-stored queries allows the user to store frequently used queries in a library and to directly reference them at execution time.
3. Modifiable logic pre-stored queries allow the user to temporarily alter query logic prior to execution.
4. Parameterized pre-stored queries allow the user to store skeletal queries within the system and then insert parameters at execution time.
5. The cue-response format provides the user with the capability to engage in a question/answer dialogue with the system. The user furnishes answers to the system questions in order to execute his request.
6. The prompting format is one which is designed primarily for inexperienced users. The system prompts the user when he is uncertain of how to formulate a request.
7. The programmable format allows the user to program and execute retrieval requests.
8. The specification of each of these functions is entirely dependent upon the type of query processing envisioned by the installation.
9. A Boolean query allows the user to perform logical retrieval statements using the comparators: equal, not equal, greater than, less than, less than or equal to, greater than or equal to. Additionally, the user can combine statements by the logical connectors AND, OR, and NOT. Thus, the user can perform such functions as: Search for "this AND this" but "NOT this"; search for "this OR this" but "NOT this"; etc.
10. A quantitative query allows the user to retrieve records based upon a count of the unique occurrences of a specified value.
11. An arithmetic query provides the user with the capability to perform arithmetic operations on data values and to retrieve based upon the numeric result.
12. Statistical query allows the user to retrieve records based upon statistical calculations such as derivation from mean, median, etc.
13. Application-defined query processing are those queries required for special applications.
14. This function allows the user to construct a qualification form based upon a previous computation or query.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-------------|--------------|----------|
| | | Initial | Extended |
| 1.3.4 DATA OUTPUT | | | |
| (a-c) The following methods of report control must be supported by the system: <ul style="list-style-type: none"> (a) user structured, (b) system defined, (c) interactively defined. | 1 | | |
| (d) The system must provide pre-stored report formats for data output. | 2 | | |
| (e-i) The system must support the following media: <ul style="list-style-type: none"> (e) CRT, (f) typewriter, (g) printer, (h) magnetic tape drive, (i) card punch. | 3 | | |
| (j) The system must provide support for multiple copy facilities. | 4 | | |
| (k-m) The system must provide the following types of output labeling: <ul style="list-style-type: none"> (k) headings, (l) trailers, (m) data labels. | 5 | | |
| (n-o) The system must provide the following data formatting capabilities: <ul style="list-style-type: none"> (n) horizontal/vertical positioning, (o) right/left justification. | 5 | | |
| (p-q) The system must provide the following data altering capabilities: <ul style="list-style-type: none"> (p) data editing, (q) decoding. | 5 6 7 | | |
| (r-s) The system must provide the following output accounting capabilities: <ul style="list-style-type: none"> (r) counting, (s) totaling. | 5 | | |
| (t-u) The system must provide pagination control for the following: <ul style="list-style-type: none"> (t) printed reports, (u) CRT displays. | 8 | | |

1.3.4 DATA OUTPUT

Reference:

1. User structured reports provide the installation with the greatest flexibility over the report format. System defined reports, while less likely to produce a format to completely satisfy every report requirement, are normally considerably faster. System defined reports are usually adequate for printing transaction listings and other internally oriented documents. Externally oriented documents (those for other than the actual DMS programmer) should probably be produced via a user structured report.

Interactively defined reports should, of course, only be considered for facilities supporting on-line interactive support.
2. This criterion is highly desirable for a data management system that will process standard reports on a recurring basis.
3. The selection of any of these criteria is dependent upon the devices available in the hardware configuration.
4. This function is a highly desirable feature for a data management system in that it provides the user with the capability to obtain multiple copies without rerunning the output program.
5. Most data management systems provide these capabilities.
6. The capability of the system to automatically perform data editing is very important to a user in producing an acceptable report appearance. Editing consists of such things as suppressing leading zeroes, supplying punctuation, etc.
7. If the DMS supports data encoding when data is entered into the file, then the output phase should support a corresponding decoding module.
8. If the system is to support such devices as printers or CRTs then pagination control should be provided for starting a new page, numbering sequential pages, stopping output on one page and beginning a new page, etc.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|---------------------------|--------------|----------|
| | | Initial | Extended |
| <p>2.1.1 UNFORMATTED DISPLAY</p> <p>(a-d) Unformatted display must be provided by the system for the following media:</p> <ul style="list-style-type: none"> (a) card, (b) magnetic tape, (c) paper tape, (d) random access storage. | 1 | | |
| <p>2.1.2 FORMATTED DISPLAY</p> <p>(a-b) The system must allow format specification by:</p> <ul style="list-style-type: none"> (a) system, (b) user. <p>(c-d) The system must support the following special display categories:</p> <ul style="list-style-type: none"> (c) data files, (d) directories. <p>(e) The system must provide for selective record display.</p> <p>(f) The system must provide for selective field display.</p> | 1 2 3 4 4 | | |

2.1.1 UNFORMATTED DISPLAY

Reference:

1. If the unformatted display option has been selected as a criterion (see 2.1 b), then it should be provided for each storage medium within the system.

2.1.2 FORMATTED DISPLAY

Reference:

1. This criterion is highly desirable since it gives the user the capability to specify the output presentation, e.g., octal, decimal, alpha op-codes, etc.
2. The capability to display data files in a formatted mode is extremely useful to the user during program testing and debugging. It is also useful to the installation as a method for displaying total file contents. This type of file display frequently becomes a backup to the report generation capabilities of a data management system.
3. If the system maintains directories or catalogs, then the system should also provide routines to produce formatted displays of their contents.
4. This criterion gives the user the capability to visually examine portions of a file without requiring a total file dump. This can be a definite aid during system troubleshooting or program testing.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|---|--------------|----------|
| | | Initial | Extended |
| <p>2.2.1 VOLUME POSITIONING</p> <p>(a-c) The system must support the following types of volumes:</p> <ul style="list-style-type: none"> (a) magnetic tape, (b) direct access storage, (c) card files. <p>(d-h) The system must provide the following magnetic tape support:</p> <ul style="list-style-type: none"> (d) forward spacing, (e) backspacing, (f) rewinding, (g) unloading, (h) erasing. <p>(i-j) The system must provide selective positioning of:</p> <ul style="list-style-type: none"> (i) files, (j) records. <p>(k-l) The system must support the following methods of positioning:</p> <ul style="list-style-type: none"> (k) count of records or files, (l) field or record contents. | <p>1</p> <p>2</p> <p>3</p> <p>3</p> <p>3</p> <p>4</p> | | |

2.2.1 VOLUME POSITIONING

Reference:

1. The specification of these criteria is highly dependent upon the hardware configuration.
2. These criteria should be specified for all processing systems which support magnetic tape.
3. This criterion is highly desirable for all processing systems supporting sequential access devices.
4. This method frequently occurs in systems supporting direct access storage.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|--|--------------|----------|
| | | Initial | Extended |
| <p>2.2.2 MEDIA COPY FACILITIES</p> <p>(a-f) The system must provide copy facilities for the following media:</p> <ul style="list-style-type: none"> (a) card, (b) magnetic tape, (c) paper tape, (d) random access storage, (e) main storage, (f) remote terminal devices. <p>(g-j) The system must provide the following options:</p> <ul style="list-style-type: none"> (g) field insertion, (h) field selection, (i) format conversion, (j) code conversion. <p>(k-m) The system must provide the following dump and reload facilities:</p> <ul style="list-style-type: none"> (k) file compaction, (l) storage compaction, (m) backup file creation. | <p>1</p> <p>2</p> <p>3</p> <p>3</p> <p>4</p> | | |

2.2.2 MEDIA COPY FACILITIES

Reference:

1. This criterion should be specified for all media supported by the processing system.
2. Field insertion and field selection provides the user with the compatibility to alter or copy selected data fields rather than entire records.

Format and code conversion is generally not too extensive in utility programs. It is normally only a conversion of data structure rather than a conversion of the data itself. Data code conversion, however, is necessary when the media use a different code representation e.g., punched card to paper tape.

3. These considerations normally apply only to direct access storage devices. File and storage compaction is highly desirable when the file is fairly volatile in the number of records maintained. As records are deleted from a file, "holes" of unused space occur. Compaction eliminates these unused areas, thus decreasing the total amount of space required for file storage.
4. Backup file creation is always a useful feature in case of inadvertent file destruction.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-------------------|--------------|----------|
| | | Initial | Extended |
| <p>2.2.3 DATA EDITING FACILITIES</p> <p>(a-c) The following types of editing must be provided by the system:</p> <p>(a) full file compare, (b) selective field compare, (c) single file scan.</p> <p>(d) The system must provide for file comparison between differing file formats.</p> | <p>1</p> <p>2</p> | | |
| <p>2.2.4 TEST DATA FILE SUPPORT</p> <p>(a-b) The system must support the creation of the following types of test files:</p> <p>(a) data files, (b) I/O terminal message files.</p> <p>(c) The system must provide history/trace file interpretation.</p> | <p>1</p> <p>2</p> | | |

2.2.3 DATA EDITING FACILITIES

Reference:

1. A full file compare is useful in validating file contents. A selective field compare is useful in validating selected field updates without validating the entire file. A single file scan is useful in checking file structure, sequence, and formats.
2. This capability occurs in very few processing systems. It can, however be quite useful when validating related files.

2.2.4 TEST DATA FILE SUPPORT

Reference:

1. Many systems provide utilities to support the checkout of application programs in a mode which will not impact operational files. For this to be done it is necessary to create a simulated environment as close to the operational environment as possible. By providing test data files and terminal message files, application program testing becomes considerably easier.
2. This criterion is highly desirable for a processing system that provides trace capabilities (see Part I, 3.2.2).

3.4.4 Fourth Level of Detail (Part III - Data Manipulation Functions)

This subsection covers the following fourth level data manipulation functions:

- 1.1.2.1 FILE SECURITY CONTROL
- 1.1.2.2 READ/WRITE ACCESS CONTROL
- 1.1.2.3 CONCURRENT ACCESS CONTROL
- 1.2.1.1 SEQUENTIAL ACCESS CONTROL
- 1.2.1.2 KEYED/INDEXED ACCESS CONTROL
- 1.2.1.3 TELEPROCESSING ACCESS CONTROL
- 1.3.2.1 STRUCTURE DEFINITION
- 1.3.2.2 SPACE ALLOCATION
- 1.3.2.3 INPUT TRANSACTION PROCESSING
- 1.3.2.4 LOGICAL RECORD MAINTENANCE
- 1.3.2.5 INTERACTIVE FILE MAINTENANCE CONTROL
- 1.3.2.6 FILE REORGANIZATION

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| <p>1.1.2.1 FILE SECURITY CONTROL</p> <p>(a-b) The system must provide the following file security control protection methods:</p> <ul style="list-style-type: none"> (a) user ID, (b) passwords. | 1 | | |
| <p>1.1.2.2 READ/WRITE ACCESS CONTROL</p> <p>(a-b) The system must provide the following access restrictions to authorized users:</p> <ul style="list-style-type: none"> (a) read access only, (b) read and selective write access. <p>(c) The system should provide unrestricted access to authorized users.</p> | 1 | | |
| <p>1.1.2.3 CONCURRENT ACCESS CONTROL</p> <p>(a-c) The system must provide the following level of data sharing:</p> <ul style="list-style-type: none"> (a) file, (b) record, (c) data element. | 1 | | |

1.1.2.1 FILE SECURITY CONTROL

Reference:

1. If the user ID is used, then the system must maintain a checklist of the users that can access a particular file and perform a user ID compare to determine if the user can have access. If the password method is used, then each file has a designated password and the mere reference by a user to the password allows access. Therefore, the password method is more efficient from a processing standpoint, though potentially less secure than user ID control.

1.1.2.2 READ/WRITE ACCESS CONTROL

Reference:

1. Once a user has been granted access to a file, some systems limit the type of operation he may perform. For example, some users may only read the file, other's may read the file and alter existing records, but may not add new records. The type of access designations that should be permitted must be derived from the anticipated file contents and the type of users requiring access.

1.1.2.3 CONCURRENT ACCESS CONTROL

Reference:

1. Concurrent access control is required to maintain the scheduling and handling of concurrent or simultaneous requests for a data file from separate programs or users in a multiprogramming or time-sharing environment. Basically, the control function must determine if multiple user access can be permitted or whether the file must be restricted to single user access. In situations where multiple users may simultaneously access a single file, it is usually desirable to grant any number of them read-only access but to restrict write access to a single user at a time.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|--|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.1.1 SEQUENTIAL ACCESS CONTROL | | | |
| (a) The system must provide basic sequential access control. | | | |
| (b) The system must provide automatic read-ahead sequential (queued) access. | 1 | | |
| 1.2.1.2 KEYED/INDEXED ACCESS CONTROL | | | |
| (a) The system must provide automatic key computation. | 1 | | |
| (b) The system must provide automatic index maintenance. | 2 | | |
| (c) The system must allow multiple index levels to be maintained. | 3 | | |
| (d-e) The system must support the following types of access keys: | | | |
| (d) software keys, | | | |
| (e) hardware keys. | | | |

1.2.1.1 SEQUENTIAL ACCESS CONTROL

Reference:

1. This function is usually desirable for sequential record processing. Automatic read-ahead facilities are provided to decrease the amount of wait time a program may have to incur during successive access operations. In this way the program can be processing previously obtained data while the next data access is being performed.

1.2.1.2 KEYED/INDEXED ACCESS CONTROL

Reference:

1. Automatic key computation relieves the user of determining the physical storage address of the record.
2. Automatic index maintenance relieves the user of updating the index of an indexed file when he adds or deletes a record from the file.
3. Indexes are provided for more efficient access of data. However, if the index itself becomes too large, then an index to the index may be desirable. For example, a disk index could consist of an index for all cylinders in a file and a separate track index for all tracks on a cylinder.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.2.1.3 TELEPROCESSING ACCESS CONTROL | | | |
| (a) The system must provide automatic message time stamping. | 1,2 | | |
| (b) The system must provide optional message time stamping. | 2 | | |
| (c-e) The system must provide the following input message processing facilities: | 3 | | |
| (c) message routing, | | | |
| (d) message queuing, | | | |
| (e) message priority recognition. | | | |
| (f-h) The system must provide the following output message processing facilities: | 3 | | |
| (f) message routing, | | | |
| (g) message queuing, | | | |
| (h) message priority recognition. | | | |

1.2.1.3 TELEPROCESSING ACCESS CONTROL

Reference:

1. This criterion is highly desirable for all systems supporting teleprocessing.
2. Message time-stamping is one of the more convenient ways of attaching a unique identifier to each message entering the system. At the same time, it allows the system to easily recognize messages that have been in the system an excessive length of time in order to increase their processing priority if necessary.
3. These features are quite desirable for most processing systems supporting teleprocessing.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.2.1 STRUCTURE DEFINITION | | | |
| (a-e) The system must allow the following types of structures: | 1 | | |
| (a) sequential, | 2 | | |
| (b) hierarchical, | 3 | | |
| (c) indexed, | 4 | | |
| (d) ring, | 5 | | |
| (e) list. | 5 | | |
| (f-g) The system must provide the following types of indexing schemes: | | | |
| (f) normal, | | | |
| (g) inverted. | 6 | | |
| (h-j) The system must allow the following types of data elements: | | | |
| (h) fixed length, | | | |
| (i) variable length, | | | |
| (j) repetitive. | 7 | | |
| 1.3.2.2 SPACE ALLOCATION | | | |
| (a-d) The system will provide data management system support using the following storage media: | 1 | | |
| (a) tape, | | | |
| (b) disc, | | | |
| (c) drum, | | | |
| (d) mass storage. | | | |

1.3.2.1 STRUCTURE DEFINITION

Reference:

1. While frequently employed for evaluation, these criteria are rarely specified. However it may be necessary to specify certain of these criteria based upon anticipated file design requirements.
2. A sequential structure is one in which the data elements are all of equal rank. This method can be used in support of data which can be grouped into data elements that are an entity within themselves.
3. Hierarchical structures provide the capability to structure a file based on a ranking scheme usually related to a specific type of logical group relationship. An example of this type of data could be a file containing a logical ranking such as: nation, political subdivision, counties, major cities, etc.
4. The indexed structure may be applied to either the sequential or hierarchical structure method and can be used to selectively locate data elements within a file.
5. The ring and list type of file structure are closely related. Each data element contains the address of the next successive data element within a file. The difference in the two structures is that the last data element in a ring contains the address of the first data element, whereas the list does not. A ring structure thus allows the user to read a total sequence of data elements starting with any element within the ring.
6. This criterion is highly desirable for a processing system where the retrieval elements are either random or unpredictable. In this structure retrieval time is minimized regardless of the data fields queried. However, this type of structure also requires an extremely complex file updating technique.
7. Repetitive data elements are those which have a number of entries under a single data label. For example, in a bank account record, deposits and/or withdrawals tend to be repeating entries.

1.3.2.2 SPACE ALLOCATION

Reference:

1. The specification of these criteria is dependent upon the hardware configuration.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-------------------------------------|--------------|----------|
| | | Initial | Extended |
| <p>1.3.2.3 INPUT TRANSACTION PROCESSING</p> <p>(a-c) The system must provide input transaction processing facilities that allow:</p> <ul style="list-style-type: none"> (a) input data definition, (b) input data validation, (c) input data alteration. <p>(d-e) The system must allow the following types of input formats:</p> <ul style="list-style-type: none"> (d) pre-established, (e) self-defining. <p>(f-i) The system must provide the following types of input data validation:</p> <ul style="list-style-type: none"> (f) equal value comparison, (g) range verification, (h) masked comparison, (i) sequence checking. <p>(j-l) The system must provide the following types of input data alteration:</p> <ul style="list-style-type: none"> (j) automatic truncation/padding, (k) encoding/decoding, (l) constant factor modification. <p>(m-o) The system must recognize the following input data terminator:</p> <ul style="list-style-type: none"> (m) standard (embedded) field, (n) special character(s), (o) physical media markers. | <p>1</p> <p>2</p> <p>3</p> <p>4</p> | | |

1.3.2.3 INPUT TRANSACTION PROCESSING

Reference:

1. These criteria should be specified for all data management systems.
2. In many cases the structure of the input to be received can be established in advance. At other times the method in which the user receives the input precludes effective data structuring.

The pre-established format is useful for fixed record input media such as punched cards, magnetic tape, etc. The self defining format is usually best for teleprocessing based transactions.

3. The ability to truncate or pad data is useful in forcing data conformity. Encoding can be useful in decreasing storage requirements, while decoding is the reverse but allows the user to make abbreviated references. Constant factor modification allows input data values to be modified by a data value.
4. Input terminators are used to signal the end of the input data to be processed. Each of these techniques has its advocates, and no particular method is favored.

| OPERATING SYSTEM REQUIREMENTS CHECKLIST | Reference | REQUIREMENTS | |
|---|-----------|--------------|----------|
| | | Initial | Extended |
| 1.3.2.4 LOGICAL RECORD MAINTENANCE | | | |
| (a) The system must allow the selection of update records by logical query. | 1 | | |
| (b) The system must provide record maintenance through multi-record logic. | 2 | | |
| (c) The system must provide automatic updating of subordinate files. | 3 | | |
| 1.3.2.5 INTERACTIVE FILE MAINTENANCE CONTROL | | | |
| (a-b) The system must provide interactive override capabilities for: | 1 | | |
| (a) data values, | | | |
| (b) update logic. | | | |

1.3.2.4 LOGICAL RECORD MAINTENANCE

Reference:

1. Logical query provides the user the capability to update records based upon a specified condition being satisfied. An example of this would be the statement "delete all records after a given calendar date" or "retain all records between two given calendar dates".
2. Update by multi-record logic provides the user the capability to update records by referencing another record or records as control.
3. A system that provides subordinate files should have the capability to automatically update these files when an update is performed on the master file to which they are subordinate.

1.3.2.5 INTERACTIVE FILE MAINTENANCE CONTROL

Reference:

1. These criteria are highly desirable for a data management system that supports interactive maintenance from on-line terminals. This feature provides a user with the capability to override established data validation conditions or data definition in an on-line interactive mode.

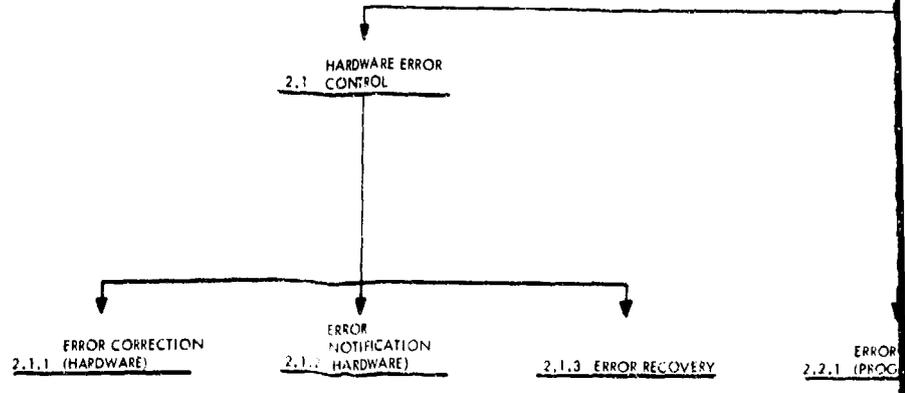
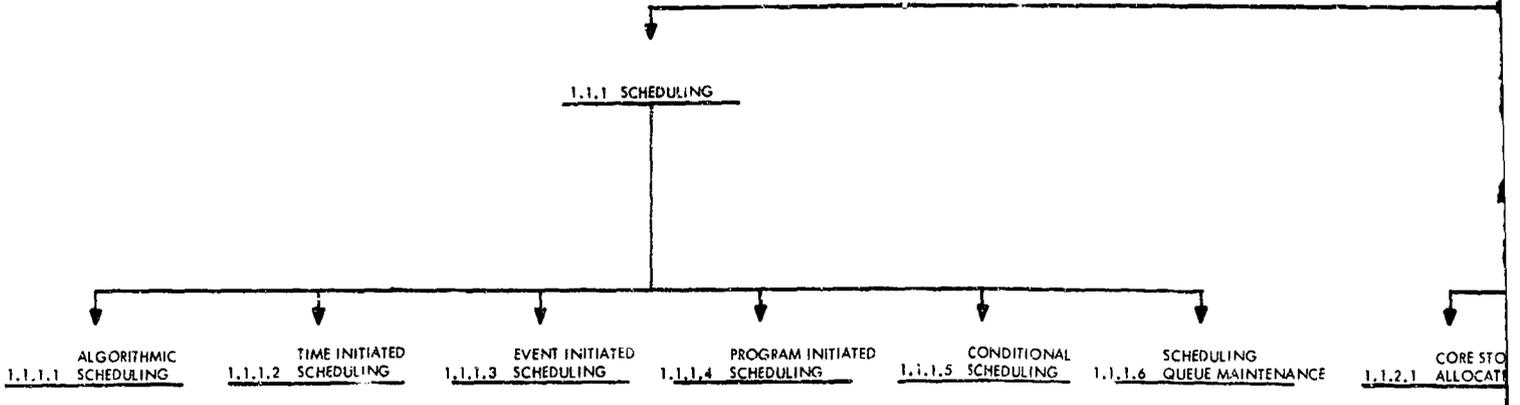
1.3.2.6 FILE REORGANIZATION

Reference:

1. These methods are frequently found in data management systems and are desirable for meeting varying file design requirements.
2. Intra file merging involves the changing of the internal structure of a file by restructuring and merging records within the file.

Inter-file merging consists of merging different files into a single file structure which will better satisfy a particular application. This allows the user to build master files of related data from existing singularly used files.

A



B

1.1 JOB CONTROL

RESOURCE
1.1.2 ALLOCATION

PROGRAM
1.1.3 LOADING

CORE STORAGE
ALLOCATION

I/O DEVICE
1.1.2.2 ALLOCATION

COMMON ROUTINE
1.1.2.3 ALLOCATION

LOADING
1.1.3.1 CONTROL

SWAPPING
1.1.3.2 CONTROL

STRUCTURE
1.1.3.3 CONTROL

DISPATCHING
1.1.4.1 CONTROL

EVENT
1.1.4.2 SYN

DIAGNOSTIC ERROR
2.0 PROCESSING

PROGRAM ERROR
2.2 CONTROL

INTERFACE ERROR
2.3 CONTROL

ERROR CORRECTION
2.2.1 (PROGRAM)

ERROR NOTIFICATION
2.2.2 (PROGRAM)

PROGRAM
2.2.3 TERMINATION

OPERATOR KEY-IN
2.3.1 EDITING

CONTROL
2.3.2 COMMAND EDITING

REMOTE TERMINAL
2.3.3 COMMUNICATION
EDITING

OPERATING SYSTEM
1.0 MANAGEMENT

PROGRAM
2.0 MAINTENANCE

OPERATING SYSTEM FUNCTIONAL CLASSIFICATION STRUCTURE

PART I - EXECUTIVE/CONTROL FUNCTIONS

1.0 JOB MANAGEMENT

C

1.2 I/O CONTROL

EVENT MONITORING
1.1.4

PROGRAM TERMINATION PROCESSING
1.1.5

1.2.1 I/O SCHEDULING

1.2.2 DATA TRANSFER

DEVICE MANIPULATION
1.2.3

WATCHDOG CONTROL

EVENT SYNCHRONIZATION
1.1.4.2

INTERRUPT PROCESSING CONTROL
1.1.4.3

PROGRAM LIMIT MONITORING
1.1.4.4

BUFFERING CONTROL
1.2.2.1

DATA CODE TRANSLATION
1.2.2.2

TELETYPE TERMINAL MANIPULATION

PROGRAM TO SYSTEM LINK
2.3.4 VERIFICATION

REAL-TIME CLOCK SERVICE
3.1.1

INTERVAL TIMER SERVICE
3.1.2

STORAGE DUMP CONTROL
3.2.1

TESTING DEBUG SERVICE
3.2.2 TRACKING

PART II - SYSTEM MANAGEMENT FUNCTIONS

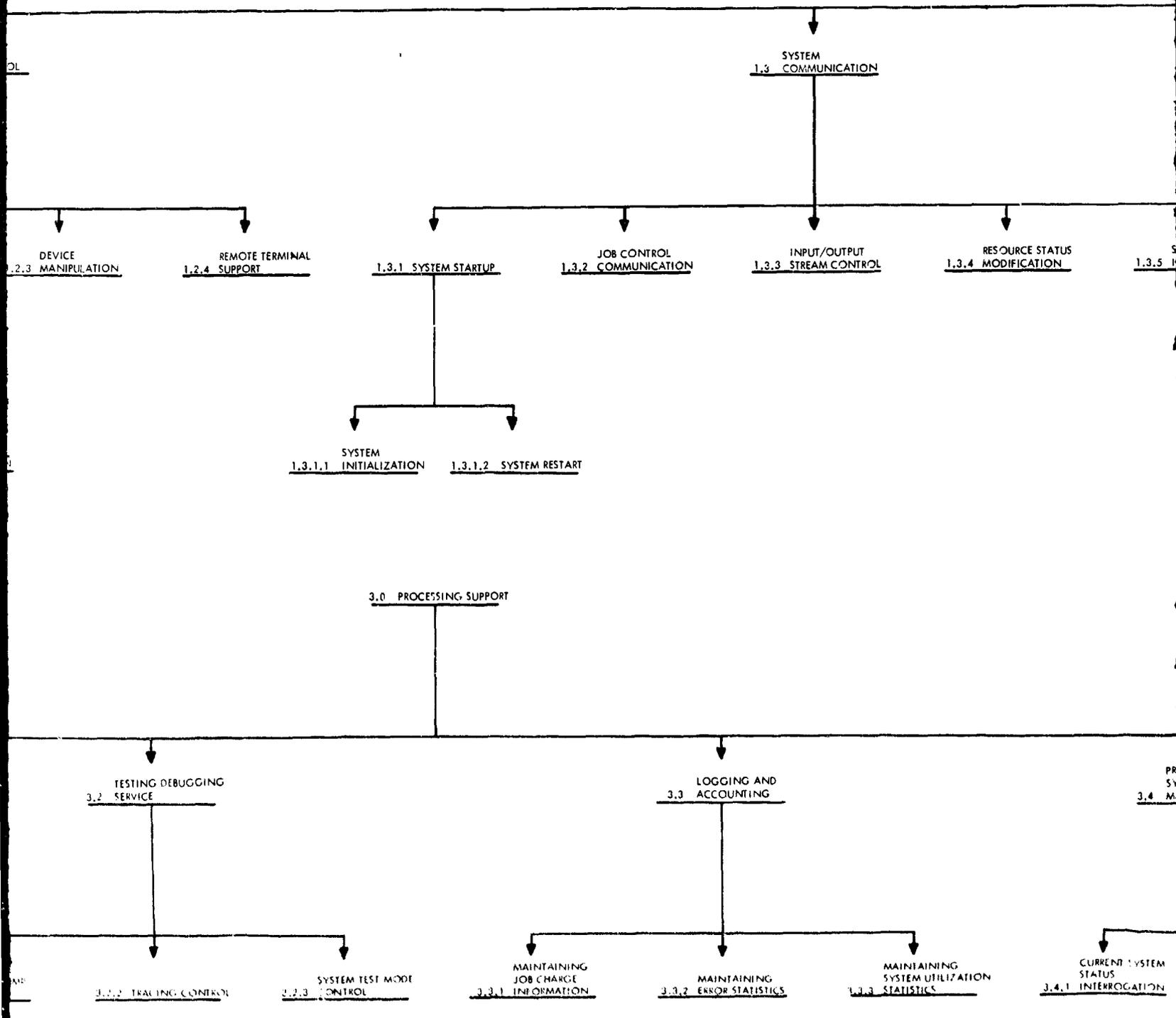
PROGRAM MAINTENANCE

3.0 COMPILER INTERFACES

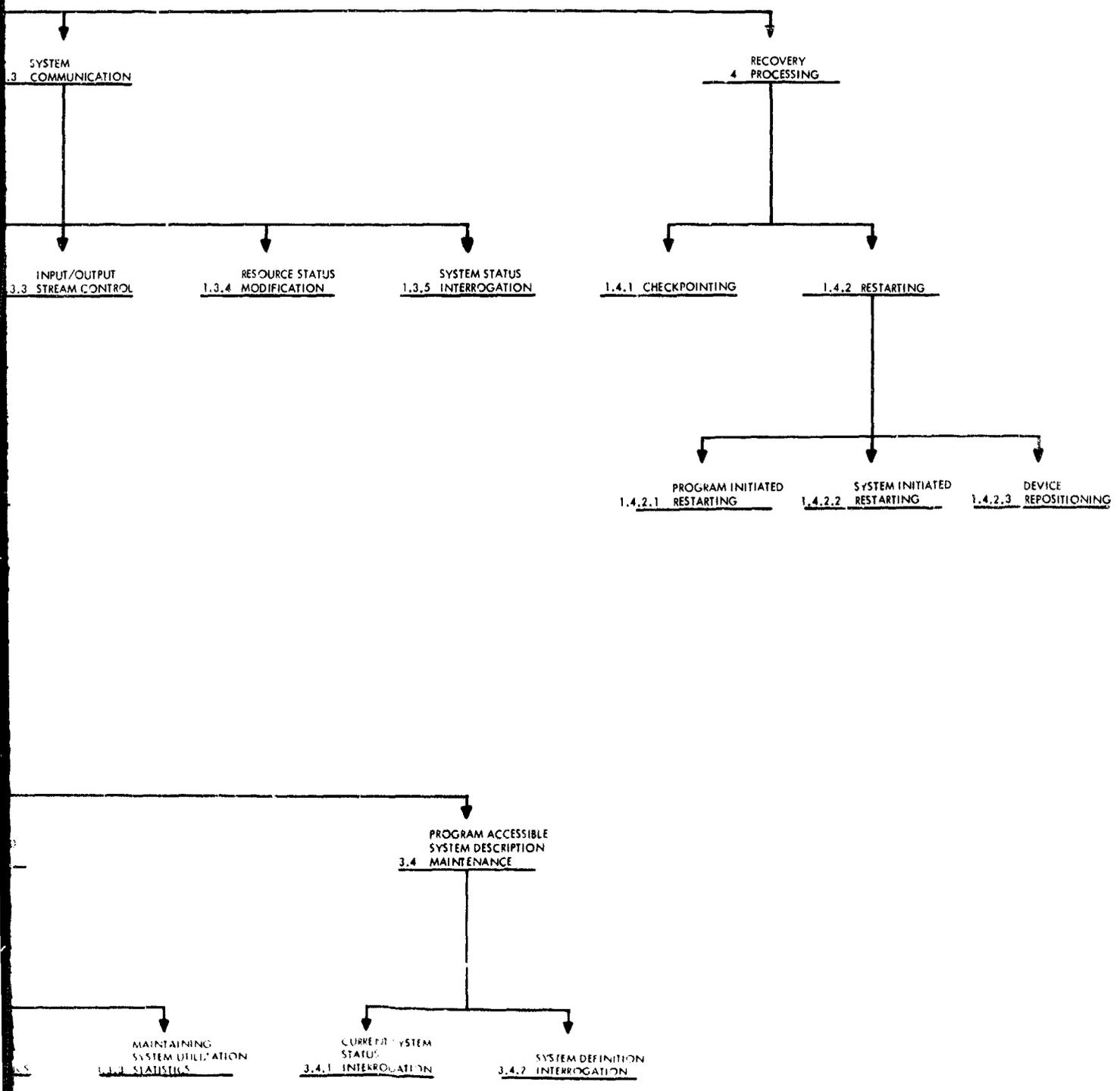
MANAGEMENT SUPPORT UTILITIES
4.0

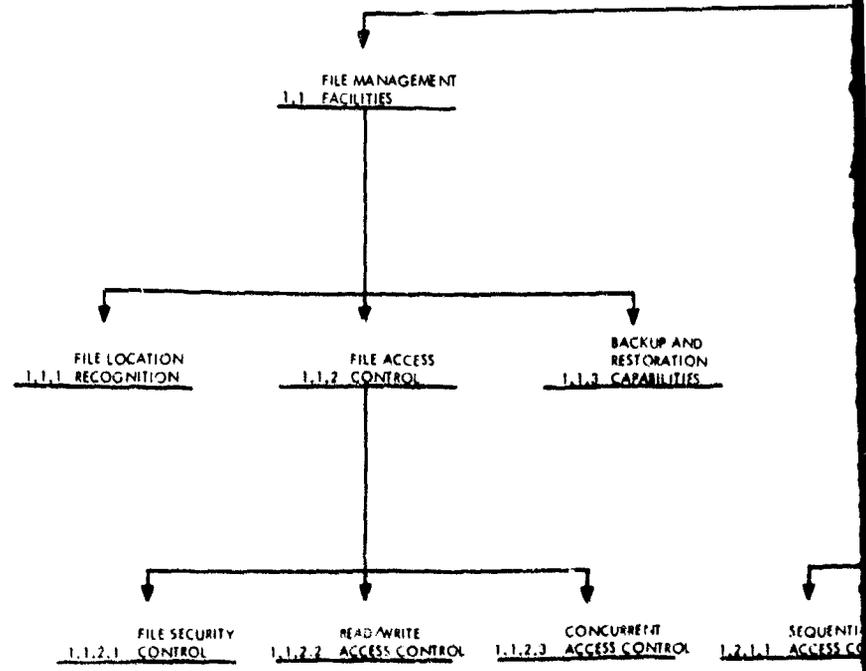
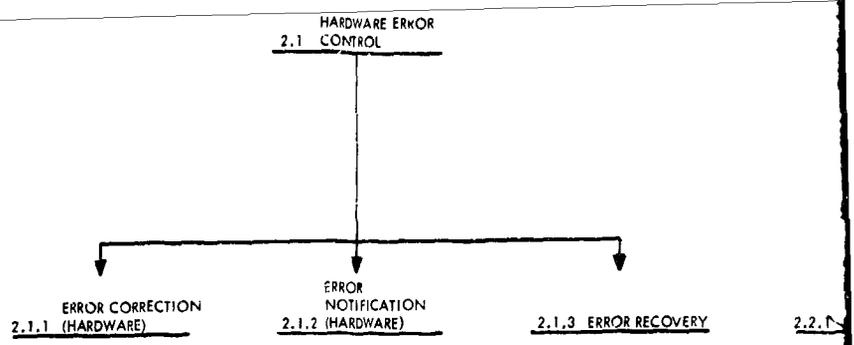
STRUCTURE

D



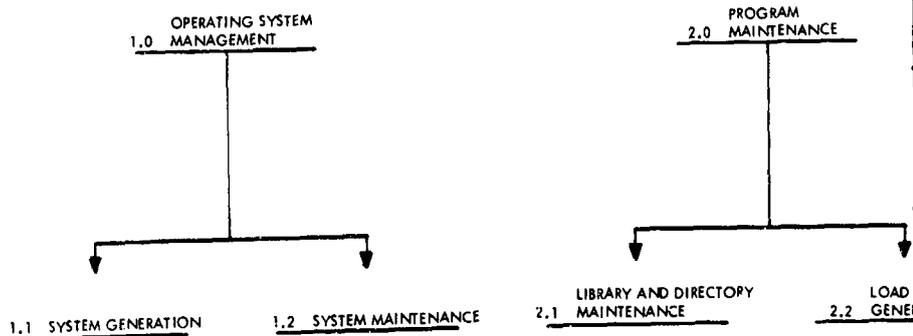
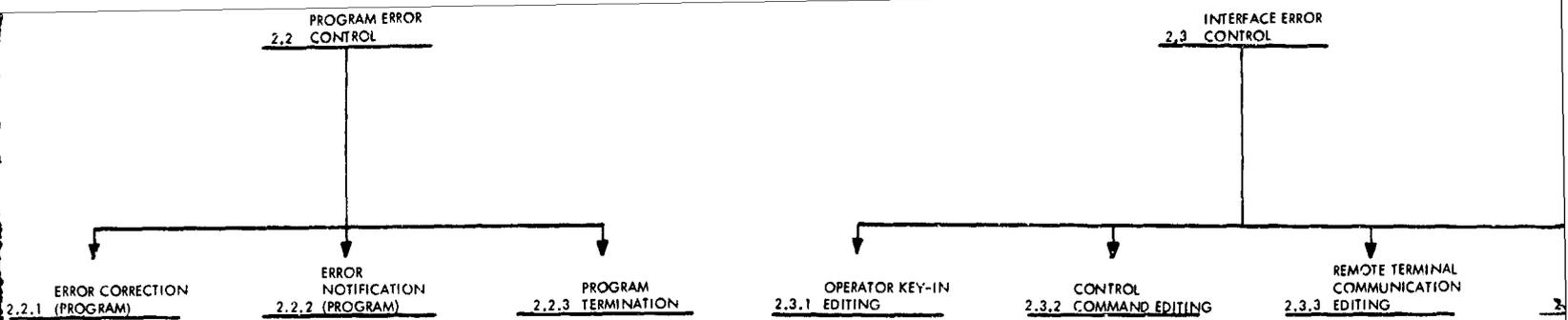
E



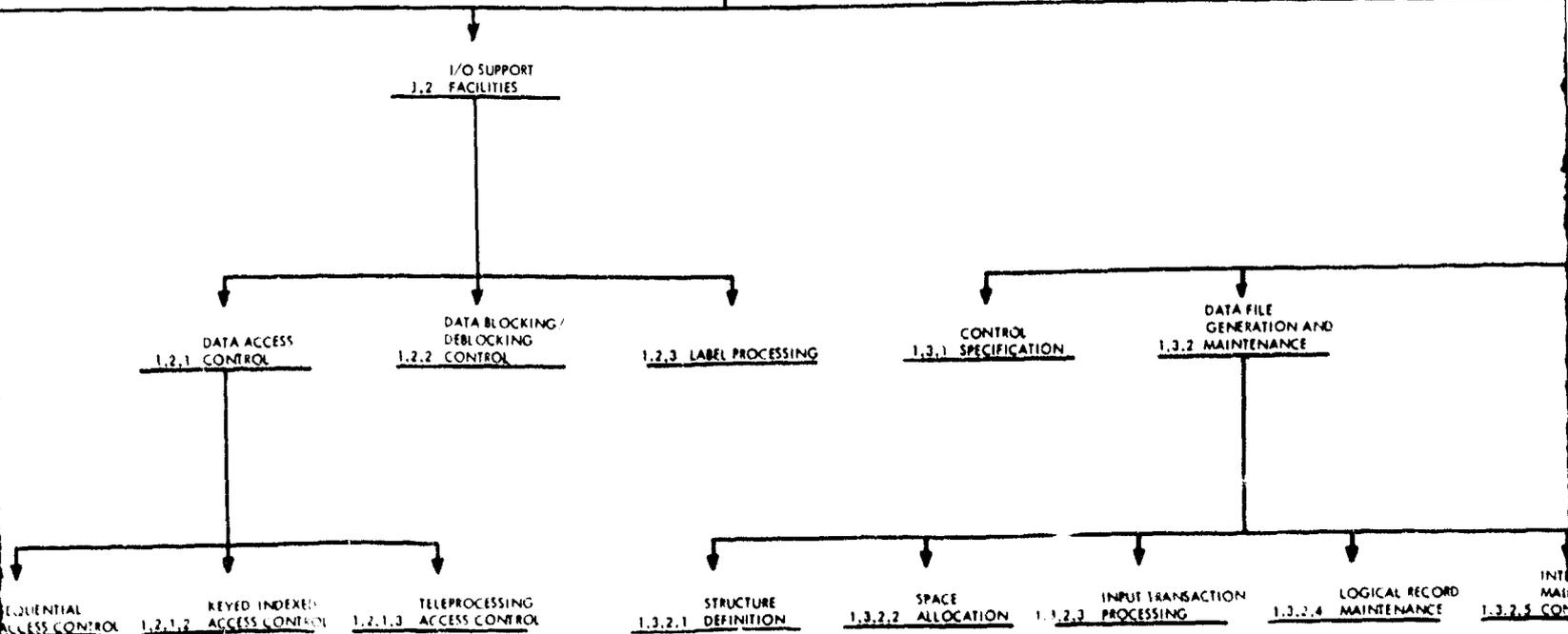


Prepared by The COMTRE Corporation, Coral Gables, Florida,
for the Electronic Systems Division (ESD), Air Force Systems
Command, under Contract Number : 19628-70-C-0258.

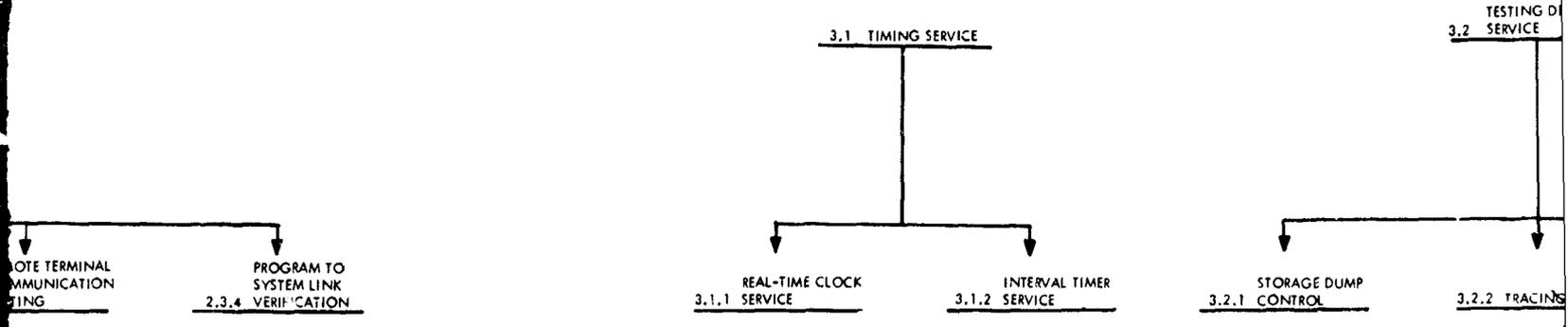
F



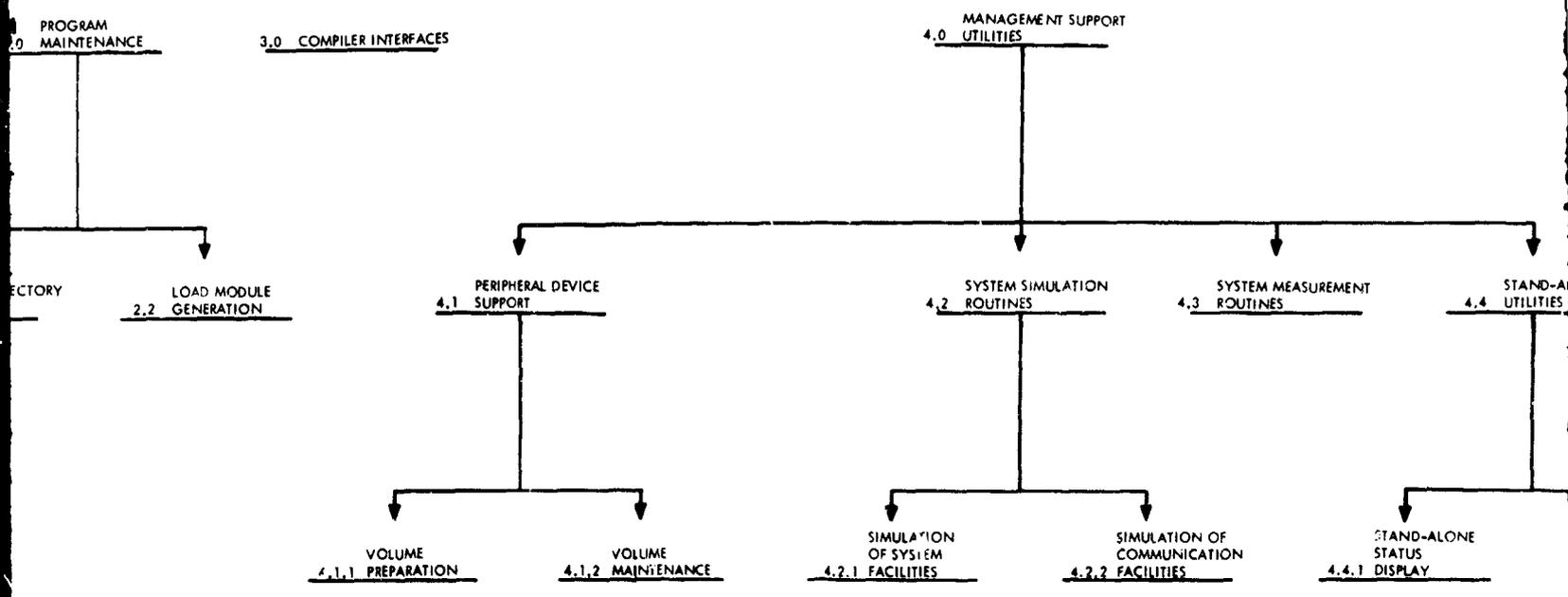
1.0 DATA MANAGEMENT



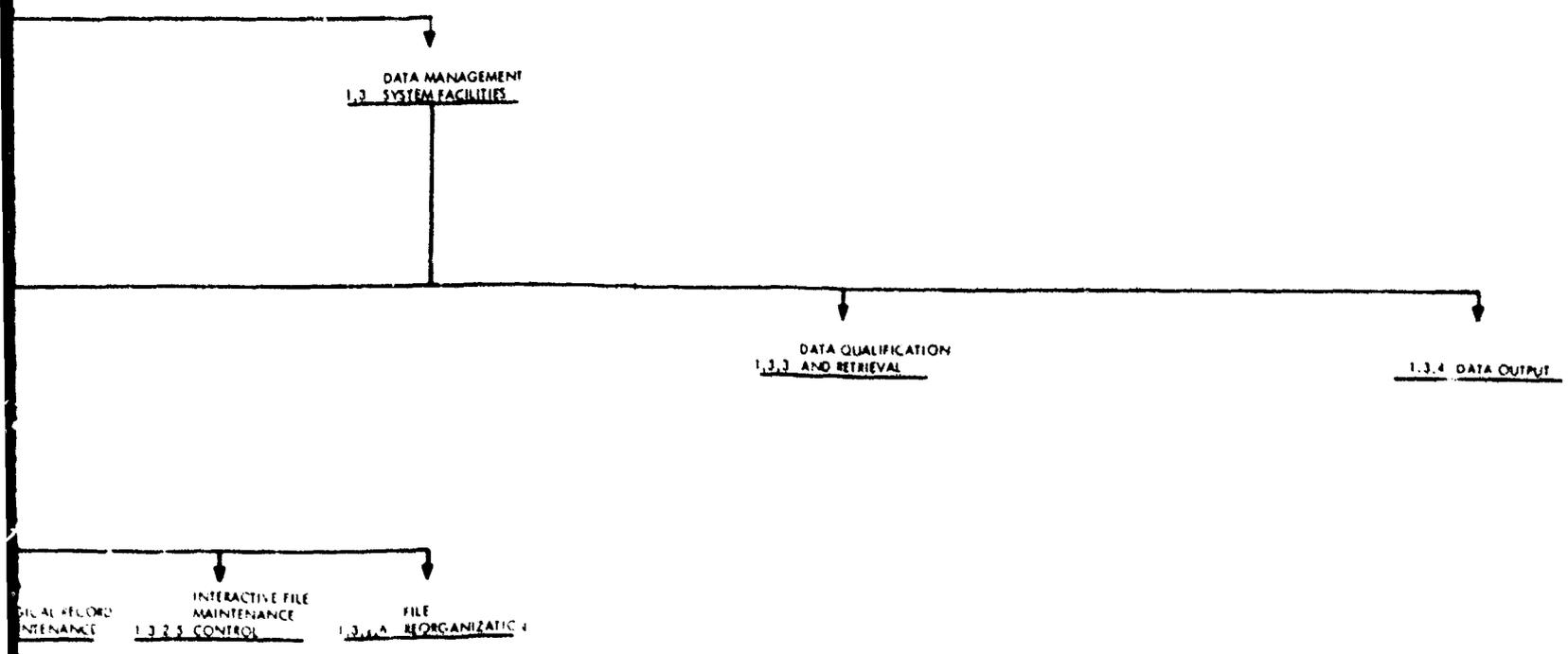
G



PART II - SYSTEM MANAGEMENT FUNCTIONS



PART III - DATA MANIPULATION FUNCTIONS



H

TESTING DEBUGGING
3.2 SERVICE

LOGGING AND
3.3 ACCOUNTING

3.2.7 TRACING CONTROL 3.2.3 SYSTEM TEST MODE CONTROL 3.3.1 MAINTAINING JOB CHARGE INFORMATION 3.3.2 MAINTAINING ERROR STATISTICS 3.3.3 MAINTAINING SYSTEM UTILIZATION STATISTICS 3.4.1 CURRENT SYSTEM STATUS INTERROGATION

STAND-ALONE UTILITIES
4.4

4.4.1 STAND-ALONE STATUS DISPLAY 4.4.2 STAND-ALONE RECOVERY SUPPORT

DATA HANDLING
2.0 UTILITIES

2.1 DISPLAY FACILITIES

PERIPHERAL DEVICE
2.2 SUPPORT

3.1.1 REV

1.3.4 DATA OUTPUT

UNFORMATTED
2.1.1 FACILITIES

FORMATTED
2.1.2 DISPLAY

VOLUME
2.2.1 POSITIONING

MEDIA COPY
2.2.2 FACILITIES

DATA EDITING
2.2.3 FACILITIES

I

END
S.G.

PROGRAM ACCESSIBLE
SYSTEM DESCRIPTION
3.4 MAINTENANCE

ING
STICS

MAINTAINING
SYSTEM UTILIZATION
3.3.3 STATISTICS

CURRENT SYSTEM
STATUS
3.4.1 INTERROGATION

SYSTEM DEFINITION
3.4.2 INTERROGATION

ING

3.0 SORTING AND MERGING

PERIPHERAL DEVICE
3.2 SUPPORT

3.1 SORT MODULE DEVELOPMENT

3.2 SORT MODULE EXECUTION

MEDIA CIPHER
3.2.2 FACILITIES

DATA LISTING
3.2.3 FACILITIES

TEST DATA FILE
3.2.4 SUPPORT

J

Attachment

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

| | | | |
|---|--|--|-------------------------|
| 1. ORIGINATING ACTIVITY (Corporate author) The COMTRE Corporation 151 Sevilla Avenue Coral Gables, Florida 33134 | | 2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED | |
| | | 2b. GROUP N/A | |
| 3. REPORT TITLE COMPUTER OPERATING SYSTEMS CAPABILITIES; A SOURCE SELECTION AND ANALYSIS AID | | | |
| 4. DESCRIPTIVE NOTES (Type of report and inclusive dates) None | | | |
| 5. AUTHOR(S) (First name, middle initial, last name) William C. Mittweide | | | |
| 6. REPORT DATE November 1970 | | 7a. TOTAL NO. OF PAGES 249 | 7b. NO. OF REFS None |
| 8a. CONTRACT OR GRANT NO. F19628-70-C-0258 | | 9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TR-71-74 | |
| b. PROJECT NO 6917 | | 9b. OTHER REPORT NO(S) (Any other numbers that may be associated with this report) | |
| c. | | | |
| d. | | | |
| 10. DISTRIBUTION STATEMENT This document has been approved for public release and sales; its distribution is unlimited. | | | |
| 11. SUPPLEMENTARY NOTES | | 12. SPONSORING MILITARY ACTIVITY Deputy for Command and Management System Hq Electronic Systems Division (AFSC) L G Hanscom Field, Bedford, Mass. 01730 | |
| 13. ABSTRACT <p>This report presents a method for translating operational data processing requirements into specific criteria for use in selecting, validating or evaluating computer operating systems. The criteria have been structured on the basis of an integrated functional classification structure applicable to the executive/control functions, system management functions and data manipulation functions of currently available operating systems. In concert with the methodology presented, a checklist form is included as an aid to developing selection criteria for particular applications. A diagram of the functional classification structure is also included.</p> | | | |

DD FORM 1473

Security Classification

| 14 KEY WORDS | LINK A | | LINK B | | LINK C | |
|--|--------|----|--------|----|--------|----|
| | ROLE | WT | ROLE | WT | ROLE | WT |
| operating systems (computer) specification evaluation validation analysis selection guidelines identification | | | | | | |