27
7-11-76
sp. & NTIS

UCID- 17198

Lawrence Livermore Laboratory

A PARTIALLY ANNOTATED BIBLIOGRAPHY

FOR COMPUTER PROTECTION AND RELATED TOPICS

Jeffrey C. Huskamp

July 20, 1976

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

A Partially Annotated Bibliography for Computer Protection and Related Topics

Jeffrey C. Huskamp

Introduction

The following sections provide references for the commonly cited technical papers in the area of computer protection. Great care is taken to exclude papers with no technical content or merit. For the purposes of this bibliography, computer protection is broadly defined to encompass all facets of the protection problem. The papers cover, but are not limited to, the topics of protection features in operating systems (e.g. MULTICS and HYDRA), hardware implementations of protection facilities (e.g. Honeywell 6180, System 250, BCC 5000, B6500), data base protection controls, confinement, and protection models.

Computer protection is not an isolated topic but relates to many other areas in computer science and electrical engineering. For this reason, a bibliography of related areas is included after the protection bibliography. These sections also include articles of general interest in the named areas which are not necessarily related to protection.

This bibliography is intended to be an ongoing document. As new articles of interest are published, they will be included provided they are of sufficient technical merit and related to computer protection. Revisions of this document can be obtained from the ELF directory:

.426075:FBIBLIO

and may be printed using the TRIX PRINT command.

Any comments on the accuracy or scope of this bibliography are welcomed.

PROTECTION

Anderson, James P.
"Information Security in a Multi-User Computer Environment," *Advances in Computers*, Volume 12, Academic Press, New York, (1972), pp. 1-36.

Survey article on the different areas of security. Discusses techniques for control of who accesses the computer (password and user identification schemes), hardware support of multi-user systems (base and bounds, read/write protect, privileged/unprivileged mode), a poor survey of file access control, incomplete design, program identity, superficial models of sharing, and encription. The major usefulness of this article is for stimulating ideas about related topics that may have been overlooked during the initial investigation of some aspect of security. The depth on any one topic is shallow.

Andrews, G. R.
COPS - A Mechanism for Computer Protection, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 5-25.

COPS (COoperation, Privacy and Sharing) is a kernel (like HYDRA) which controls the mapping from logical to physical objects. A basic monitor tells whether an access is allowed. User defined monitors may be invoked by the basic monitor to implement arbitrary access rules. Information structures are distinct from the memory that contains the representations so that the mapping may be changed. Changing of a mapping is on an all-or-nothing basis. User defined monitors are invoked when indirect capabilities are used. Confinement is implemented for overt channels by preventing the changing of objects in the incarnation segment and by preventing any other procedures from being called. There is no provision for blocking covert channels. Provides for create, grant, ungrant and destroy operations for information structures. Able to control access to physical and logical objects by separating out information structures.

Arden, B. W., et. al.
Program and Addressing Structure in a Time-Sharing Environment, *Journal of the ACM*, 13:1 (January 1966), pp. 1-16.

Motivates program relocation and base registers. An addressing technique similar to MULTICS is presented. The segment number /

page number / offset are arbitrary divisions of the address. A connection region is used to hold symbolic addresses. When a new segment is bound at execution time, its generalized linkage segment containing symbolic addresses is copied into the connection region. The symbolic addresses are transformed into segment number, etc. on first use (delayed binding). This paper is valuable mainly for historical purposes and for a slightly different view of linkage segments than MULTICS.

Barton, R. S.
A New Approach to the Functional Design of a Digital Computer, *Proceedings of the Western Joint Computer Conference*, (1961), pp. 393-396.

This paper outlines the major features of the B5000 machine including the Polish notation arithmetic evaluation scheme and the subroutine linkage. Advocates designing 'special purpose' computers, such as the B5000 to facilitate programming. The importance of this paper is as a historical reference.

Baskin, Herbert B., et. al.
PRIME - A Modular Architecture for Terminal-Oriented Systems, *Proceeding of the Spring Joint Computer Conference*, 40 (1972), pp. 431-437.

Overview of the goals and initial implementation of PRIME. Attacked problems of hardware/software reliability, system availability (dynamic reconfiguration), system extendability (adding processors or memory) and user specified operating systems. The control monitor oversees all operations and runs continuously on one processor. ECM's (local monitors) execute in each subsystem to handle user I/O, user traps, etc. The control monitor handles system-wide events and the ECM's handle local user events. This organization is very close to one needed for confinement. However the division of responsibility between the control monitor and the ECM's needs to be more specific and with a more definite orientation to the confinement problem.

Belady, L. A. and C. Weissman
Experiments With Secure Resource Sharing for Virtual Machines, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 27-33.

Surveys the joint effort of IBM and SDC to penetrate VM/370. The flaw detection methodology included documentation of possible flaws, information flow graphing to determine control objects, filtering by thought experiments of low probability/low payoff hypotheses, desk checking and live testing. VM was found to be penetratable only by utilizing channel programs running asynchronously with the CPU. Suggests correcting flaws by making a VM dispatchable only if there are no outstanding I/O requests. This article has value as a documentation of a successful penetration attempt and as a testimonial to the securability of the VM architecture.

Bensoussan, A. and C. T. Clingen and R. C. Daley
The MULTICS Virtual Memory: Concepts and Design, *Communications of the ACM*, 15:5 (May 1972), pp. 308-318.

Presents arguments for MULTICS addressing design choices and gives a detailed account of how the addressing is done. Points out that sharing must be done without duplication of information. Gives reasons for segmentation and paging.

Bisbey, R. and G. Popek
Encapsulation: An Approach to Operating System Security, *ISI/RR-73-17*, USC/Information Sciences Institute, (October 1973).

Discusses using a minicomputer to flush one operating system out of a machine and to load another. The mini would also control access to peripheral devices by setting switches on the devices depending on what operating system is executing. Hopes to make mini code small enough to be proven correct. The use of VMM's (or hypervisors) is discounted due to the cost of complex I/O simulation. This varies with the results of Belady and Weissman. This idea automates what the military is used to doing.

Bobrow, D. G. and B. A. Wegbreit
A Model and Stack Implementation of Multiple Environments, *Communications of the ACM*, 16:10 (October 1973), pp. 591-603.

Presents a technique for arbitrary retention of procedure activation blocks on a stack. Each block consists of a fixed sized basic frame holding the parameters and a frame extension containing activation storage. The frame extension has a pointer to the next higher entries in the dynamic (call) chain and the static variable binding resolution chain. Coroutines can be implemented.

Bobrow, D. G., et. al.
TENEX, A Paged Time Sharing System for the PDP-10, *Communications of the ACM*, 15:3 (March 1972), pp. 135-143.

Overview of the structure of TENEX. Modified PDP-10 with the BBN pager. Sharing of pages in memory and copy-on-write provided. Well human engineered. File access based on whether the program's attached directory is the same as the owning directory, whether the program's attached directory is in the same group as the owning directory or neither of the above. Details the scheduler.

Branstan, Dennis K.
Privacy and Protection in Operating Systems, *Computer*, 6:1 (January 1973), pp. 43-46.

Report of a workshop sponsored by the IEEE committee on operating systems. Includes short summaries of project SUE, CAP, PRIME and Rotenberg's model.

Burroughs Corporation
The Descriptor - A Definition of the B5500 Information Processing System, *Bulletin 5000-20002-P*, Detroit, Michigan, (February 1961).

Describes the B5000 machine. The most interesting aspects are the push-down operand stack and the use of descriptors. There is one program reference table per process which contains descriptors and constants. Programming can only be done in a higher level lenguage. There is an addressing problem for nested ALGOL blocks.

Buzen, J. P. and U. Gagliaridi
The Evolution of Virtual Machine Architecture, *Proceedings of the National Computer Conference*, 42 (1973), pp. 291-299.

Survey paper on the development of virtual machines. Distinguishes between type 1 VM's which are not able to be invoked recursively and type 2 VM's which are. Mentions single state VM's where each level allocates a subset of its address space to the next level. Two types of faults: software visible (privileged/non-privileged) and VM specific (attempt to alter a resource map maintained by the VMM).

Carlstedt, Jim and Richard Bisbey II and Gerald Popek
Pattern Directed Protection Evaluation, *ISI/RR-75-31*, USC/Information Sciences Institute, (June 1975).

Suggests automating the search for protection errors by describing a generic set of protection errors (patterns) then writing a program to process the system listings and to find instances of the pattern. Suggests there are less than 25 types of errors. Raw errors are generalized and parameterized to be applicable to a wide range of systems. The general description is then made system specific. Feature extracting is the hard part. Claims that was able to apply the method to MULTICS for finding one type of error.

Carroll, J. M., et. al.
Multi-Dimensional Security Programs for a Generalized Information Retrieval System, *Proceedings of the Fall Joint Computer Conference*, 39 (1971), pp. 571-577.

Gives the design of a toy data base manager. A user's access is governed by the password he uses on access to a file. Access can be controlled down to the item level. This is implemented by a bit vector associated with each password. This is not a very interesting system from the perspective of the current state of the art.

Cleary, J. G.
Process Handling on Burroughs B6500, *Proceedings of the Fourth Australian Computer Conference*, Adelaide, Australia, (1969), pp. 231-239.

This paper is an excellent summary of addressing on the B6500. Details stack usage for process creation, stack activation/deactivation and display registers. Other topics that are not of as much interest include locking, memory protection, event implementation and interrupts.

Cohen, Ellis, et. al.
HYDRA User's Manual, Internal Paper, Carnegie-Mellon University, (November 1974).

Explains the details of the HYDRA implementation. Includes the defined kernel calls available. Confinement is implemented as absolute confinement.

Cohen, Ellis and David Jefferson
Protection in the HYDRA Operating System, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 141-160.

Defines a negotiated decision as one user wanting to limit the restrictions imposed by another user (e.g. freezing an object). Many of HYDRA's protection mechanisms are implemented by capability rights. All objects except data objects have both a C-list and a data part. Procedure objects use the C-list to contain the templates and own capabilities and the data part to contain the code. Explains type-type capability and type object capabilities. The procedure call mechanism is explained. States that protection problems can be solved directly by implementing a mechanism or by procedural embedding. Discusses the solution (or attempted solution) of the problems: mutual suspicion, protection against modification, limit capability propagation, revocation of parameters on subsystem return (conservation), confinement (storage channels), initialization, revocation, guaranteed access (freezing), accounting problem, lost object problem (vague). Best generally available description of HYDRA.

Conn, Richard W. and Richard H. Yamamoto
A Model Highlighting the Security of Operating Systems, *Proceedings of the ACM National Conference*, (1974), pp. 174-179.

Argues that existing systems can be represented by a graph model with nodes representing states of the computation (contents of registers and memory cells) and with arcs representing operations. Assumes all systems can be broken down into hierarchical levels. Extraneous nodes can be eliminated by using folding.

Conn, Richard W.
Flow Models for Operating System Security, Internal Paper, Lawrence Livermore Laboratory, (1974).

A slightly more readable version of the paper presented at ACM '74. It includes an example of folding not found in the other paper. The argument that every system is hierarchical is unconvincing.

Conway, R. W. and W. L. Maxwell and H. L. Morgan
On the Implementation of Security Measures in Information Systems, *Communications of the ACM*, 15:4 (April 1972), pp. 211-220.

Distinguishes between privacy and security. The security matrix is advocated for information security. Presents ways to decrease the size of the sparse matrix (aggregation, activating subsets of the matrix). Column system has all or nothing access while the diagonal system has each user with a disjoint set of resources. Wants data independent accesses to be checked at translation time and data dependent accesses to be checked at run time. Gives ASAP as an example of a system that embodies these principles. Have problems with dynamic revocation of access.

Conway, R. W. and W. L. Maxwell and H. L. Morgan
Selective Security Capabilities in ASAP - A File Management System, *Proceedings of the Spring Joint Computer Conference*, 40 (1972), pp. 1181-1185.

Every file has an authorized user directory appended. Checking done at translation time when possible. Each record is assigned one of nine security classes. Separates actions (R,W,E) from whether access is allowed. Can break security by accessing files not through ASAP. ASAP has its own language.

Corbato, F. J. and M. Merwin-Daggett and R. C. Daley
An Experimental Time-Sharing System, *Proceedings of the Spring Joint Computer Conference*, 21 (1962), pp. 335-344.

Describes an early implementation of CTSS at MIT. Gives the motivation for developing time-sharing systems, the commands implemented and the multi-level scheduling queue devised by Corbato. Useful as a historical perspective paper.

Corbato, F. J. and V. A. Vyssotsky
Introduction and Overview of the MULTICS System, *Proceedings of the Fall Joint Computer Conference*, 27 (1965), pp. 185-196.

Contains no specifics about the proposed MULTICS but lists some goals and how MULTICS intends to solve the associated problems. Makes an extensive argument as to why time sharing is desirable. The two part addressing scheme is presented and the motivation for segments and pages is given. Design goals for the software, the file system and the I/O equipment are given. Not much substance to this paper.

Corbato, F. J. and J. H. Saltzer and C. T. Clingen
MULTICS - The First Seven Years, *Proceedings of the Spring Joint Computer Conference*, 40 (1972), pp. 571-583.

Provides a short history and current status of MULTICS. Able to change system modules while the system is running. Can provide subsystems that can look like another system. The project administrator can set the environment of a member's execution. Most of the system written in PL/1. Manages a three level storage hierarchy automatically. Good review of MULTICS' progress thru 1971.

Cosserat, D. C.
A Capability Oriented Multi-Processor System for Real-Time Applications, *Proceedings of the International Conference on Computer Communications*, Washington, D.C., (October 1972), pp. 282-289.

Discusses the Plessey 250 system for telephone switching control. Design goals are: (1) ability to add hardware as processing needs increase, (2) the system must be extremely reliable, (3) must be able to interface with varied telecommunications devices. Capabilities used for storage protection. Enter capabilities used for domain switching. Freely copyable capabilities as advocated by Fabry are implemented.

Creech, B. A.
Architecture of the B-6500, *Software Engineering*, 1 (1970), pp. 29-43.

Describes the advantages of the B6500 over the B5500. In designing the B6500 the following principles were used: need for recursion, compact representation of code and data, deferment of binding as late as possible, importance of subroutines, exclusive use of higher level languages, and use of multiprocessing. Identifies descriptor as master or copy. The copies are linked to the master but not vice-versa. Tagged memory for data in word. Problem for 'tricky' programs which use character strings as integers. In storage allocation must find all copies of a descriptor to set the 'not present' bit.

Daley, R. C. and P. G. Neumann
A General Purpose File System for Secondary Storage, *Proceedings of the Fall Joint Computer Conference*, 27 (1965), pp. 213-230.

Discusses the generally accepted tree structured file system. Links are used to point to an entry in another directory. The working directory is used to decrease the length of path names. Explains the access control list and attributes trap, read, write, execute and append. Trap causes a trap whenever the file is accessed to permit arbitrary protection algorithm. Never implemented because didn't know in whose space the trap routine should execute. Backup by incremental and weekly dumping. Storage hierarchy implemented for files based on activity. The proposed file system module implementation is given.

Daley, Robert C. and Jack B. Dennis
Virtual Memory, Processes and Sharing in MULTICS, *Communications of the ACM*, 11:5 (May 1968), pp. 306-313.

Hierarchical file system specifications. Trap attribute not implemented for lack of proper domain to execute in. Access control list associated with each branch implements security. File system participates in MULTICS addressing as a repository for certain information. File back-up scheme presented which includes incremental dumping.

Denning, Dorothy E.
A Lattice Model of Secure Information Flow, *Communications of the ACM*, 19:5 (May 1976), pp. 236-243.

The paper is concerned with showing that flows of information (e.g. from one object or program variable to an ther) is permissable. A painful derivation of a lattice structure is done and the lattice properties are shown to be intuitively correct for security. Explicit flows of information (e.g. assignment statements) are shown to be rather easily checked. Implicit flows for static bindings are resolved by a high-water-mark type solution. Dynamic bindings are harder in that security violations may be used to transmit information. Solution used is to reset the security classification of all objects referenced in the conditional statement based on the conditional variable.

Denning, D. E. and P. J. Denning and G. S. Graham
Selectively Confined Subsystems, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 55-61.

Contends that a subsystem which is called with confidential data can not be allowed to leak any un-confidential data without the possibility existing that confidential information may be disclosed. Thus selective confinement of information is not viable.

Dennis, J. B.
Programming Generality, Parallelism, and Computer Architecture, *Proceedings of IFIP 1968*, North Holland, Amsterdam, (1968), pp. C1-C7.

Dennis, J. B.
Segmentation and the Design of Multiprogrammed Computer Systems, *Journal of the ACM*, 12:4 (October 1965), pp. 589-602.

Dennis, J. B. and E. C. Van Horn
Programming Semantics for Multiprogrammed Computations, *Communications of the ACM*, 9:3 (March 1966), pp. 143-155.

Spheres of protection defined by C-lists. Superior/inferior spheres used for debugging. Protected entry used to enter protection sphere of callee (callee is given a suspended process capability for the caller). Capabilities can only be dispersed by the owner. Mutually suspicious can be performed with a dummy interface routine. A debugger can debug a new version of the debugger if the owned inferior sphere capability for the object program (calling sequence old debugger - new debugger - object program) is placed in the directory so the old debugger can access it.

Endres, Albert
An Analysis of Errors and Their Causes in System Programs, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 327-336.

Classifies errors uncovered during internal IBM testing of DOS/VS (release 28). About 500 modules were affected (169 new, 253 patched, 100 comment change only). Found 432 errors after new version had been integrated and made operational. 52 percent of the modules had no errors and 27 percent had one error. 46 percent of the errors occurred due to lack of complete understanding of the problem (all possible cases, etc.). 38 percent could be helped by better methods and tools. A list of measures to be taken to help remedy the errors is given.

England, D. M.
Architectural Features of System 250, *INFOTECH State of the Art Report on Operating Systems*, (1972).

England, D. M.
Capability Concept Mechanisms and Structure in System 250, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 63-82.

England, D. M.
Operating System of System 250, *Proceedings of the International Switching Symposium*, Boston, Massachusetts, (June 1972), pp. 525-529.

Evans, Arthur, Jr. and Edwin Weiss
A User Authentication Scheme Not Requiring Secrecy in the Computer, *Communications of the ACM*, 17:8 (August 1974), pp. 437-442.

Presents a hard-to-invert function which is derived from the password. This means that the password table and the function do not need to be kept secret. The method used is a nesting of functions where the number of times each function is applied is dependent on the original password and the result from the last function. At least one of the functions must be non-linear.

Evans, D. C. and J. Y. LeClerc
Address Mapping and Control of Access in an Interactive Computer, *Proceedings of the Spring Joint Computer Conference*, 30 (1967), pp. 23-30.

Fabry, R. S.
A User's View of Capabilities, *ICR Quarterly Report*, Institute of Computer Research, University of Chicago, Chicago, Illinois, :15 (November 1967), pp. IC-1-8.

Discusses the Chicago Magic Number Computer when the design of the system was just begun. Divides the resources into directly accessible (memory), supervisor accessible (teletypes) and pseudo-accessible (extended types). Was concerned about segments which have been rolled out to disk.

Fabry, R. S.
Capability-Based Addressing, *Communications of the ACM*, 17:7 (July 1974), pp. 403-412.

Fabry, R. S.
Dynamic Verification of Operating System Decisions, *Communications of the ACM*, 16:11 (November 1973), pp. 659-668.

Fabry, R. S.
Preliminary Description of a Supervisor for a Machine Oriented Around Capabilities, *ICR Quarterly Report*, Institute of Computer Research, University of Chicago, Chicago, Illinois, :18 (August 1968), pp. i b1-b97.

Fenton, J. S.
Memoryless Subsystems, *The Computer Journal*, 17:2 (May 1974), pp. 143-147.

Ferrie, J., et. al.
An Extensible Structure for Protected Systems Design, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 83-105.

Feustel, Edward A.
On the Advantages of Tagged Architecture, *IEEE Transactions on Computers*, 7 (July 1973), pp. 644-656.

Feustel, Edward A.
The Rice Research Computer - A Tagged Architecture, *Proceedings of the Spring Joint Computer Conference*, 40 (1972), pp. 369-377.

Friedman, T. D.
The Authorization Problem in Shared Files, *IBM Systems Journal*, 9:4 (1970), pp. 258-280.

Gaines, R. S.
An Operating System Based on the Concept of a Supervisory Computer, *Communications of the ACM*, 15:3 (March 1972), pp. 150-156.

Goldberg, R. P.
Architecture of Virtual Machines, *Proceedings of the National Computer Conference*, 42 (1973), pp. 309-318.

Graham, G. Scott and Peter J. Denning
Protection - Principles and Practice, *Proceedings of the Spring Joint Computer Conference*, 40 (1972), pp. 417-429.

Graham, R. M.
Protection in an Information Processing Utility, *Communications of the ACM*, 11:5 (May 1968), pp. 365-369.

Explanation of access bracket and software support needed The function of gates, the method of performing inter-ring calls and the method of argument validation are discussed. Pertains to GE 645.

Gray, J., et. al.
The Control Structure of an Operating System, *IBM Research Report RC 3949 (#17842)*, San Jose, California, Restricted distribution, (July 1972).

Halton, D.
Hardware of the System 250 for Communication Control, *Proceedings of the International Switching Symposium*, Boston, Massachusetts, (June 1972).

Hamer-Hodges, K. J.
Fault Resistance and Recovery Within System 250, *Proceedings of the International Conference on Computer Communication*, Washington, D.C., (October 1972), pp. 290-296.

Hansen, Per Brinch
RC4000 Software Multiprogramming System, *RCSL No: 55-D140*, A/S Regnecentralen, Copenhagen, (February 1971).

The RC4000 has a 24-bit word with instruction execution time of 4 microseconds. Structured into a monitor and a subordinate tree of processes. The root of the tree is a system process which initially holds all the resources. No swapping is done but the paper suggests that it could be done. Seems that the monitor is invoked for many functions, including handling the file system. Messages are implemented by buffers which are sent to a destination then are returned as answers. I/O performed by sending messages to the handlers. Problem with finite number of buffers. In order to obtain immediate initiation of I/O, the monitor is non-interruptable for long periods of time. Sounds like the 'external processes' are also in the kernel. Parent processes have absolute control (modify control) over child

processes. New operating systems (parents) are easily introduced. Easy to debug new version of an operating system.

Hansen, P. B.
The Nucleus of a Multiprogramming System, *Communications of the ACM*, 13:4 (April 1970), pp. 238-241+.

Harrison, Michael A. and Walter L. Ruzzo and Jeffrey D. Ullman
On Protection in Operating Systems, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 14-24.

A protection system is modeled as Lampson's access matrix. All protection-related operations are modeled as changes in the access matrix. A system is safe for right R if there does not exist a configuration of subjects, objects and the access matrix such that right R is potentially leaked. Concludes there is no algorithm which can decide if an arbitrary configuration of an arbitrary system, or of all configurations of a given system is safe. Must consider specific cases for protection verification.

Hartley, D. F. and B. Landy and R. M. Needham
The Structure of a Multiprogramming Supervisor, *The Computer Journal*, 11:3 (November 1968), pp. 247-255.

Discusses the supervisor for the Titan (Atlas 2) computer. Base and limit register design with addresses being "ORed" into the base register address. Programs can only start at certain places in the memory. Space allocated to user programs may be used by the supervisor until the user claims the space (pre-emption). Protection provided by three program counters: main control (user mode with no privileges), extracode control (supervisor), interrupt control (for interrupt programs - all interrupts are inhibited). Describes I/O (the well), scheduler and job queueing.

Hauck, E. A. and B. A. Dent
Burroughs B6500/B7500 Stack Mechanism, *Proceedings of the Spring Joint Computer Conference*, 32 (1968), pp. 245-251.

Haugeland, W. and Bruce Lindsay and D. Redell
An Integrated Access Control Facility, Private Communication.

Hoffman, Lance J.
The Formulary Model for Flexible Privacy and Access Controls, *Proceedings of the Fall Joint Computer Conference*, 39 (1971), pp. 587-601.

Holt, Richard C. and Marc S. Grushcow
A Short Discussion of Interprocess Communication in the SUE/360/370 Operating System, Proceedings of the ACM SIGPLAN-SIGOPS Interface Meeting on Programming Languages - Operating Systems, *SICPLAN Notices*, 8:9 (September 1973), pp. 74-78.

The process tree implements a hierarchical ordering which is used for interprocess communication. A process can only communicate (via facility calls) with an ancestor. The tree of ancestors is scanned until one is found which implements the needed facility. Easy to layer the system and to augment or change facility call meanings. Bad for implementing global functions (must be pushed into the nucleus), mutually suspicious processes can not interact directly but only thru a trusted third program, since caretakers are used for data base manipulation, all data base managers which might be called must be in the process ancestry before execution, and the scheme is inflexible.

Iliffe, J. K.
Basic Machine Principles, American Elsevier, New York, (1968).

Janson, Philippe A.
Dynamic Linking and Environment Initialization in a Multi-Domain Process, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 43-50.

The dynamic linker was broken up into three parts. Two parts execute outside the security kernel (the dynamic linker and dynamic domain initializer) and one part inside (dynamic domain generator). The system model used for explanation is MULTICS-based except that multiple domains per process are allowed. The dynamic linker executes in the caller's domain to resolve who the callee is. The dynamic domain initializer

executes in the callee's domain and initializes the linkage segment for the called procedure. The dynamic domain generator creates and initializes (e.g. the words needed for the conventions) working storage (linkage segment and stack segment).

Jodeit, J.
Storage Organization in Programming Systems, *Communications of the ACM*, 11:11 (November 1968), pp. 741-746.

Discusses the addressing of the Rice computer which was designed in the early 1960's. The implementation is an extension of the B5000. Array indexing is by dope vectors. The concept of protected entry is implemented by allowing only one entry point per program (block). The entry address is in the codeword (read descriptor). Memory protection is by checking that the referenced element of the segment is within the segment.

Jones, Anita K.
Protection in Programming Systems, Ph.D. Thesis, Carnegie-Mellon University, (June 1973).

Jones, Anita K. and Richard J. Lipton
The Enforcement of Security Policies for Computation, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 197-206.

Defines a program as a function with k inputs and n outputs. A protection mechanism is a mapping where the outputs are either the same as without the protection mechanism or are from the set of violation notices. A security policy is a subset of the possible input variables. A protection mechanism is sound if the outputs can only depend on variables which are included in the security policy. Does not attempt to cover all information control policies (confinement) but at least all access control policies. Observability postulate is that all observable attributes of a program are actual outputs (not presently possible). One protection mechanism is more complete than another if both are sound and one returns fewer violations. Can union two mechanisms. The surveillance protection mechanism for flowchart programs does not allow variables outside of the security policy to be tested or assigned. The high-water-mark protection mechanism is unsound if the running time is observable and sound otherwise. A major problem is that the security policy must be mapped to a set of variables. Also there is no effective method for finding the maximal protection mechanism.

Jones, A. K. and W. A. Wulf
Towards the Design of Secure Systems, *Software - Practice and Experience*, 5 (1975), pp. 321-336.

Kilburn, T., et. al.
One-Level Storage System, *IRE Transactions on Electronic Computers*, EC-11:2 (April 1962), pp. 223-235.

Lackey, R. D.
Penetration of Computer Systems - An Overview, *Honeywell Computer Journal*, Number 2, (1974), pp. 81-85.

Lampson, B. W.
A Note on the Confinement Problem, *Communications of the ACM*, 16:10 (October 1973), pp. 613-615.

Three types of channels: storage, legitimate (e.g. the bill), covert. If a confined program call another program that is not trusted, it must also be confined. Permit caller to determine inputs into all legitimate and covert channels.

Lampson, B. W.
Dynamic Protection Structures, *Proceedings of the Fall Joint Computer Conference*, 35 (1969), pp. 27-38.

Discusses BCC Model 1 proposed capability system with domains. Attached to each capability is a string of n bits corresponding to n known domains. Bit i is on if the capability belongs to domain i. Must resolve two capabilities for the same object with different accesses. Errors processed in order of call stack. Needs to add a provision to keep an untrusted domain from getting control on an error. Gate capabilities. MULTICS ring type implementation with monitor, utility and user. Interrupts and traps.

Lampson, B. W.
On Reliable and Extendable Operating Systems, *Techniques in Software Engineering*, NATO Science Committee Workshop Material, Volume II, (same as - An Overview of the CAL Timesharing System) (September 1969), pp. .

Lampson, B. W.
Protection, *Proceedings of the Fifth Annual Princeton Conference on Information Sciences and Systems*, Princeton University, Princeton, New Jersey, (March 1971), pp. 437-443.

Presents the message system and access matrix protection models. Protected entries simulated, impersonation not possible in message system. Error handling difficult. PRIME is a message system (good covert channel containment possibilities in PRIME). Originates access matrix model. Needs provision for revocation (history of the domains granting any specific capability). A domain should be able to decrease its capability rights unilaterally.

Lampson, B. W.
"Protection and Access Control in Operating Systems," *INFOTECH State of the Art Report on Operating Systems*, C. Boon, Editor, Maidenhead, England, INFOTECH Information Ltd., (1972), pp. 309-326.

Defines context as what determines the actions that a program is authorized to do. Examples include user/supervisor state and domains. Need protected transfer between contexts. Explains the message system which is the most general from the protection point of view. The faults of the model are that one domain can not force another one to do anything (debugging) and that standard facilities for loading, compiling, etc. are needed. Naming of domains for identification is awkward since one domain does not know the names of other domains. The access matrix model (called object system) is also discussed.

Lampson, Butler W.
Redundancy and Robustness in Memory Protection, *Proceedings of IFIP 1974*, (1974).

Distinguishes between absolute protection (no matter how hard a program executing in a domain tries, it is unable to break the protection barriers of the domain) and defensive protection (protection against accidents, not malice). Each domain owns an access node in the memory hierarchy tree. Branches from the node indicate functions (fetch, store, etc.) or data. Sharing done by exchanging messages with the owner of the segment. Patterned after a message system model. Memory protection must support facilities of the overall system in system like MULTICS (e.g. revocation of access for a mapped-in file). Redundancy needed for error recovery.

Lampson, Butler W. and Howard E. Sturgis
Reflections on an Operating System Design, *Communications of the ACM*, 19:5 (May 1976), pp. 251-265.

Discusses the CAL system developed at UC Berkeley and analyzes some of the system's shortcomings. The main features are discussed but not in a very readable manner. One failure was trying to map references by software rather than hardware. The result was that the map only prevented two copies of shared information being in CM. Because of hardware limitations every data block of a referenced file had to be resident in ECS. Commonly used operations (like reading a file) were costly. Fixed sized MOT could have impaired ECS usage with more users. Automatic memory management not done since the time could not be charged to users as a consequence of their requests. Also discusses domain naming, communication and error handling. Could not represent kernel objects on the disk.

Lauer, H. C. and D. Wyeth
A Recursive Virtual Machine Architecture, *Proceedings of the ACM SIGOPS-SIGARCH Workshop on Virtual Computer Systems*, Cambridge, Massachusetts, (1973).

Outlines a recursive VM mechanism where all protection is done by the addressing mechanism. Recursive VM's are created with a subset of the resources held by the containing environment. Address resolution is indirectly thru the descriptor lists of a VM's ancestors as is done in the CAP computer. There is no supervisor state. The containing VM may set up any mapping for its descendants. Display registers as on the B6700 are used for address resolution between VM levels. A descendant may call a higher level VM to perform a service. Devices may be simulated by VM's for their descendants. Device interrupts are communicated by P and V operations. A process is immediately interrupted only if one of its ancestors is waiting for the signal.

Lauesen, Soren
A Large Semaphore Based Operating System, *Communications of the ACM*. 18:7 (July 1975), pp. 377-389.

Describes an operating system (BOSS2) which is implemented on the RC4000 using Hansen's nucleus. Coroutines are used to implement system functions due to system constraints. Proves absence of deadlock in the design. All communication done by messages and semaphores. Offers virtual memory by software paging. Discusses project time plan and gives the reason for overrun as a gross underestimate of the number of instructions needed (used 200 machine instructions per man-month, one error per 20 instructions, uncover 5 errors per test run).

Leaman, Roger J.
System 250 - Security Philosophy, *Proceedings of the Conference on Computers-Systems and Technology*, The Institute of Electronic and Radio Engineers, London, England, (October 1972), pp. 189-200.

Details the recovery mechanism for System 250. The recovery procedure for a hardware fault and the hardware testing methods are given.

LeClerc, Jean-Yves
Memory Structures for Interactive Computers, *Project GENIE Document 40.10.110*, University of California, Berkeley, (May 1966).

Levin, R., et. al.
Policy/Mechanism Separation in HYDRA, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 132-140.

The kernel provides hardware manipulation mechanisms (I/O) and parameterized policies. Policy modules (PM's) specify policy by supplying appropriate parameters to the kernel. The kernel does short-term scheduling of processes designated to execute by different policy modules on a priority basis. Paging decisions (i.e. designation of current working set) is done by user programs. The kernel makes page replacement decisions. For confinement, want to give the PM as little information as possible. Has a negotiation mechanism whereby a process under a PM can request certain treatment. If advice is not followed, an error return is made to the requesting process. Similiarly, a process can request a certain amount of contiguous time to perform a critical section. If not granted, an error return is taken.

Linde, Richard R.
Operating System Penetration, *Proceedings of the National Computer Conference*, 44 (1975), pp. 361-368.

Describes the SDC penetration methodology which consists of four parts: (1) knowledge of system control structure, (2) flaw hypothesis generation, (3) flaw hypothesis confirmation, (4) flaw generalization. Uses a security control object dependency graph as a template to see if correct access protection is done by the target operating system. This paper was criticized because (1) only a select few can be successful penetrators and therefore (2) they could find flaws without any methodology. Need to show average people how to find flaws.

Linde, R. R.
The ADEPT-50 Time-Sharing System, *Proceedings of the Fall Joint Computer Conference*, 35 (1969), pp. 39-50.

Lindsay, B. G.
Suggestions for an Extensible Capability-Based Machine Architecture, *International Workshop on Computer Architecture*, Grenoble, France, (June 1973).

Lipner, Steven B.
A Comment on the Confinement Problem, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 192-196.

Describes a method for closing the storage channels in a system based on the military classification system. Proof is bv examining the effects section of each O-function invokable by the user to make sure that the V-functions used can not depend on V-functions for higher classifications. To close covert channels wants to implement virtual time for each process. Doesn't work if the user enters timing information from the terminal at regular intervals. The virtual time solution only generates noise in this case.

Lipner, Steven B.
A Minicomputer Security Control System, *Proceedings of COMPCON74*, San Francisco, (February 1974), pp. 57-60.

Lipner, Steven B.
A Panel Session - Security Kernels, *Proceedings of the National Computer Conference*, 43 (1974), pp. 973-980.

Liskov, Barbara H.
The Design of the Venus Operating System, *Communications of the ACM*, 15:3 (March 1972), pp. 144-149.

Lobel, Jerome
The Cost of Computer Privacy, *Proceedings of the National Computer Conference*, 44 (1975), pp. 935-940.

Reports on a project funded by Honeywell to assess the cost of implementing certain privacy provisions. The merit is in the provisions listed.

Lyon, J. R.
Store and Fetch Protection, *IBM Technical Disclosure Bulletin*, 16:6 (November 1973), pp. 1844-1845.

Madnick, S. and J. Donovan
Application and Analysis of the Virtual Machine Approach to Information System Security and Isolation, *Proceedings of the ACM SIGOPS-SIGARCH Workshop on Virtual Computer Systems*, Cambridge, Massachusetts, (March 1973), pp. 210-224.

This paper reasons that VM's should be more secure than conventional operating systems for strict isolation. This view is later supported by Belady and Weissman. Points out that there is an extra security mechanism which separates VM's from each other which is not present in conventional operating systems. Uses a probability argument based on intuition to show that VM's are more secure than conventional systems. Also states that VM's usually have redundant checking of security decisions.

Mathis, Robert F.
A Panel Session - Research in Data Security - Policies and Projects, *Proceedings of the National Computer Conference*, 43 (1974), pp. 993-999.

McPhee, W. S.
Operating System Integrity in OS/VS2, *IBM Systems Journal*, 13:3 (1974), pp. 230-252.

Millen, Jonathan K.
Security Kernel Validation in Practice, *Communications of the ACM*, 19:5 (May 1976), pp. 243-250.

This paper is concerned with proving a simple security condition (if a subject has read access to an object, the security level of the object is less than or equal to the security level of the subject) and the *-property (if an untrusted subject has read access to one object and write access to another, the security level of the first must be less than or equal to the second) for a security kernel. The method divides the system into subsystems which are then proven. Must insure that each subsystem has the same security level for shared objects. Prevents storage and legitimate channels by associating security levels on each word of data in the kernel. Does not handle covert channels. An example proof is given of a function written using Parnas specifications.

Molho, Lee M.
Hardware Aspects of Secure Computing, *Proceedings of the Spring Joint Computer Conference*, 36 (1970), pp. 135-141.

Motobayashi, S. and T. Masuda and N. Takahashi
HITAC 5020 Time Sharing System, *Proceedings of the ACM National Conference*, 24 (1969), pp. 419-429.

Extends the access matrix protection model of Lampson by introducing monitors for each object type which implement the matrix for each type and by defining a revokable indirect access. The limiting concept of owner is still used. The seven levels of operating systems are explained in the introduction.

Needham, R. M. and M. V. Wilkes
Domains of Protection and the Management of Processes, *The Computer Journal*, 17:2 (May 1974), pp. 117-120.

Needham, R. M.
"Protection - A Current Research Area in Operating Systems," *International Computing Symposium*, A. Gunther and B. Levrat and H. Lipps, Editors, American Elsevier Publishing, New York (1974), pp. 123-126.

Argues for supervisor which does not need privileged state for execution. Instead uses enter capabilities for supervisor services. Limits damage that can be done. When a process can be

terminated (before completion) is not clear. What domain should be notified of an asynchronous event is not clear. Sees investigation of decentralized operating systems as a research area.

Needham, R. M. and R. D. H. Walker
Protection and Process Management in the 'CAP' Computer, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 155-160.

Supervisor calls implemented by enter capabilities which change the domain. Subsetting of resources done by capabilities in the child process indirecting thru capabilities in the parent process. Can execute copies of the operating system recursively providing that proper passage of interrupts is done. Whenever the 'kernel' is executing, it is non-interruptable. Interrupts only noticed when kernel does an enter subprocess call. Only processes with the same parent can synchronize. Enter capabilities can not be passed down the process hierarchy. Many differences between CAP and the proposed confinement system. CAP seems overly restrictive for some cases. This paper does present a good overview of the process structure in CAP.

Needham, R. M.
Protection Systems and Protection Implementations, *Proceedings of the Fall Joint Computer Conference*, 41 (1972), pp. 571-578.

Neumann, P. G., et. al.
On the Design of a Provably Secure Operating System, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 161-175.

Organick, E. I.
Computer System Organization - The B5700/B6700 Series, Academic Press, (1973).

Organick, E. I.
The MULTICS System: An Examination of Its Structure, MIT Press, Cambridge, Massachusetts, (1972).

Parnas, D. L. and D. P. Siewiorek
Use of the Concept of Transparency in the Design of Hierarchically Structured Systems, *Communications of the ACM*, 18:7 (July 1975), pp. 401-408.

Discusses the use of virtual machines to screen out undesirable states of more primitive levels. In building more sophisticated system layers, the designer must not screen out desirable functions that are available in more primitive layers. Tradeoff of transparency and flexibility of design.

Patel, R. M.
Basic I/O Handling on Burroughs B6500, *Proceedings of the Second ACM Symposium on Operating Systems Principles*, (1969), pp. 120-129.

The advantages of being able to program the I/O handler with high level primitives (events, queues) in ESPOL is demonstrated. Processes waiting on units are organized in a queue. All queue heads are collected into an array so that indexing to the right queue head can be done by the unit number. The queue manipulation routines may be user defined. I/O interrupts are handled by spawning a new process to take care of it. Processes wait for I/O completion by waiting on an event. Complex waits (the 'OR' of two or more events) are allowed.

Plessey Telecommunications Ltd.
System 250: General Introduction, *Reference SPC1*, Plessey Telecommunications Ltd., Restricted distribution, (April 1972).

Popek, Gerald J. and Charles S. Kline
A Verifiable Protection System, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 294-304.

Lists advantages of the kernel approach. The TENEX password flaw demonstrates that lower level constructs are not always successfully hidden from higher levels. Placing virtual memory support below the security kernel simplifies the kernel code but much more code to prove. The UCLA kernel is uninterruptable and doesn't fault for protection data. If the required data is not in memory, an error return is given to the requestor which must page-in the data and resubmit the request (capability faulting). Lists what functions are put in the kernel and which are excluded. Higher level 'kernels' for file management system, etc. are made from lower level kernels. Prevent sending covert information by the ordering of requests among devices by having one scheduling process per device.

Popek, Gerald J.
Protection Structures, *Computer*, 7:6 (June 1974), pp. 22-33.

Popek, Gerald J. and Charles S. Kline
Verifiable Secure Operating System Software, *Proceedings of the National Computer Conference*, 43 (1974), pp. 145-151.

Most of the contents of the article are covered in more detail in the reliability conference paper. For confinement, is worried that a scheduler which can recognize more than one type of message can implement a covert channel by the order of completion returns (Morse code problem). Solves by having one scheduler per device. Ignors the channel implemented by wait times for a shared device. In this scheme, the users are the activists and the scheduler is just a conduit for information.

Price, W. R.
Implications of a Virtual Memory Mechanism for Implementing Protection in a Family of Operating Systems, Ph.D. Thesis, Carnegie-Mellon University, (1973).

Purdy, George B.
A High Security Log-In Procedure, *Communications of the ACM*, 17:8 (August 1974), pp. 442-445.

Discusses using polynomials to a prime modulus as password encoding functions since a bound on the function degeneracy (number of passwords which map into the same encoded word) can be obtained. The probability of guessing a password depends directly on the degeneracy.

Redell, David D.
Naming and Protection in Extendable Operating Systems, *Project MAC Report MAC-TR-140*, Ph.D. Thesis, Cambridge, Massachusetts, (November 1974).

Redell, D. D. and R. S. Fabry
Selective Revocation of Capabilities, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 197-209.

Revocation done by linking map entries in the following manner: literal-copy implies occupy same node of the tree, revocable-copy implies occupy a new son node, owner occupies the root of the tree. Keep most recently resolved access rights in an associative memory. Access resolved by tracing the tree to the root node. Scheme has trouble if a capability can unilaterally decrease its rights. Can also decrease number of probes into the map to resolve the access if no revocation done to two probes.

Reed, I. S.
Information Theory and Privacy in Data Banks, *Proceedings of the National Computer Conference*, 42 (1973), pp. 581-587.

Connects information theory with encoding source records in a data base. Discusses perfect secrecy ($I(X,Y)=0$) and distortion (for statistical access). Does not present new results. Most of the paper explains concepts in information theory.

Repton, C. S.
Reliability Assurance for System 250 - A Reliable Real-Time Control System, *Proceedings of the International Conference on Computer Communications*, Washington, D.C., (October 1972), pp. 297-305.

Ritchie, Dennis M. and Ken Thompson
The UNIX Time-Sharing System, *Communications of the ACM*, 17:7 (July 1974), pp. 365-375.

Presents an abbreviated description of the features included in UNIX but not much on the internal workings of the system such as the scheduling algorithm, disk file placement strategy, etc. Introduces the concepts of pipes, filters and the Shell. One of its selling points is the versatility of the system with such low-cost hardware.

Robinson, Lawrence, et. al.
On Attaining Reliable Software for a Secure Operating System, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 267-284.

Discusses the design and proof methodology for large operating systems. Views operating systems as consisting of abstract

machines which build on the functions provided by lower levels. Permits an abstract machine to call a lower level function providing the next lower abstract machine can call it. Each module is either a V-function (value returning), an O-function (state changing), or an OV-function (indivisible), as advanced by Parnas. Gives the five stages of the proof methodology. Discusses the function of the system and it looks like a typical capability system with extended objects, etc. Illustrates the design of levels 4, 5, and 6 complete with Parnas specifications as an example.

Roder, John and A. Frederick Rosene
Memory Protection in Multiprocessing Systems, *IEEE Transactions on Electronic Computers*, 16:3 (June 1967), pp. 300-326.

Rotenberg, Leo J.
Making Computers Keep Secrets, *MIT Project MAC Report MAC-TR-115*, Ph.D. Thesis, Cambridge, Massachusetts, (February 1974).

Saltzer, Jerome H.
Protection and the Control of Information Sharing in MULTICS, *Communications of the ACM*, 17:7 (July 1974), pp. 388-402.

Describes implementation of protection after the Honeywell 6180 was delivered. Passwords, absentee batch, proxy log in, log in verification routine (optional), protected subsystems and access list implementation discussed. Weaknesses (general) of the system implementation are listed.

Saltzer, Jerome H. and Michael D. Schroeder
The Protection of Information in Computer Systems, *Proceedings of the IEEE*, 63:9 (September 1975), pp. 1278-1308.

Survey article on protection. Discusses access list systems and capability systems. No new information or insight given. Bibliography with 100 references.

Saltzer, Jerome H.
Traffic Control in a Multiplexed Computer System, *MIT Project MAC Report MAC-TR-30*, Ph.D. Thesis, Cambridge, Massachusetts, (1966).

Scherr, A. L.
Functional Structure of IBM Virtual Storage Operating Systems Part II: OS/VS2 Concepts and Philosophies, *IBM Systems Journal*, 12:4 (1973), pp. 382-400.

Scherr, A. L.
The Design of IBM OS/VS2 Release 2, *Proceedings of the National Computer Conference*, 42 (1973), pp. 387-394.

Schroeder, M. D. and J. H. Saltzer
A Hardware Architecture for Implementing Protection Rings, *Communications of the ACM*, 15:3 (March 1972), pp. 157-170.

Details ring crossing mechanism used in the new Honeywell 6180 hardware. Downward calls (trusted) and upward returns implemented in hardware (most frequent case) while converse is implemented in software. Argument validation by attaching ring number field to program registers. Gives ring usage conventions.

Schroeder, M. D.
Cooperation of Mutually Suspicious Subsystems in a Computer Utility, *MIT Project MAC Report MAC-TR-104*, Ph.D. Thesis, Cambridge, Massachusetts, (1972).

Schroeder, Michael D.
Engineering a Security Kernel for MULTICS, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 25-32.

Objective is to place the least amount of common mechanism in the kernel. If possible, mechanism should reside in user domains. Common mechanisms implement sharing, interprocess communication, and physical resource multiplexing. Philosophy is that system-provided mechanisms may contain security flaws but they are triggered by the requesting user. No outside user can exploit them. An overview of what is being done to MULTICS is given.

Sevcik, K. C. and D. C. Tsichritzis
Authorization and Access Control Within Overall System Design, *Proceedings of the International Workshop on Protection in Operating Systems*, IRIA, Paris, France, (August 1974), pp. 211-224.

Sevcik, K. C., et. al.
Project SUE as a Learning Experience, *Proceedings of the Fall Joint Computer Conference*, 41 (1972), pp. 331-339.

Spier, Michael J. and Thomas N. Hastings and David N. Cutler
A Storage Mapping Technique for the Implementation of Protective Domains, *Software Practice and Experience*, 4 (1974), pp. 215-230.

Spier, Michael J. and Thomas N. Hastings and David N. Cutler
An Experimental Implementation of the Kernel/Domain Architecture, *Proceedings of the Fourth ACM Symposium on Operating System Principles*, Yorktown Heights, (October 1973), pp. 8-21.

Distinguishes the different types of storage that are applicable to a domain based system. The types of storage are: procedure segment (shared by all processes), domain own segment (shared by all processes), incarnation own temporary segment (not shared among processes or domains - includes the stack segment), incarnation own permanent segment (preserved from one invocation to the next - not shared among processes or domains), and the process own segment (shared among all domains visited by a process - used for argument passage). An incarnation is identified by the pair (process,domain) for a domain. Implementation on a PDP-11 is discussed.

Sturgis, H. E.
A Postmortem for a Timesharing System, *Xerox PARC Technical Report 74-1*, Ph.D. Thesis, (January 1974).

Thomas, Robert H.
JSYS Traps - A TENEX Mechanism for Encapsulation of User Processes, *Proceedings of the National Computer Conference*, 44 (1975), pp. 351-360.

Describes the encapsulation implementation which was included in TENEX to allow a multi-installation file system to be implemented by the RSEXEC. The mechanism traps all system calls made by inferior processes (tree structured process hierarchy). The mechanism is recursively invokable.

Tsichritzis, D.
A Capability Based File System, Source Publication Unknown.

Presents the preliminary design of the file system for Project SUE. Two types of capabilities. Numeric capabilities can provide CPU time limit, file space limit, etc. Boolean capabilities are the usual capabilities. Describes the capability format. Uses indirection and encoding to decrease capability proliferation and to save space. File system appears to be built in and not implemented as an extended type. Keeps a linked list of active users for a file to insure two processes do not have the file open for writing at the same time. All opens, closes, creates, etc. must go through the file system. Design is messy. Capabilities created by a process eventually end up in the sponsor's directory (no lost objects).

Turn, Rein and W. H. Ware
Privacy and Security in Computer Systems, *American Scientist*, 63:2 (March-April 1975), pp. 196-203.

Describes requirements for ensuring privacy, confidentiality and security in computer systems. For privacy must ensure (1) no data base containing personal data can be kept secret, (2) individuals must be able to determine what data is recorded about them and how it is used, (3) there is a procedure for correcting or amending a record, (4) an individual consents to all usage of his data, and (5) reliability of data must be guaranteed and precautions must be taken to prevent misuse. Confidential data can be disclosed by being subpoenaed. Can use statistical methods so that subjects can answer either the sensitive question or the innocuous one and the researchers can obtain results. Can also contaminate the data files at random so statistically are the same. Can separate data from identities and keep the correspondence table in another country. A superficial summary of computer security is also given.

Vanderbilt, D. H.
Controlled Information Sharing in a Computer Utility, *MIT Project MAC Report MAC-TR-67*, Ph.D. Thesis, Cambridge, Massachusetts, (October 1969).

Vysstosky, V. A. and F. J. Corbato and R. M. Graham
Structure of the MULTICS Supervisor, *Proceedings of the Fall Joint Computer Conference*, 27 (1965), pp. 203-212.

Is a fairly general paper which outlines the initial direction for MULTICS. The file system is responsible for over half the code. The approaches to scheduling (denial of service for heavy loads rather than degradation), dynamic linking and traps are explained. Not many specifics.

Walker, R. D. H.
The Structure of a Well-Protected Computer, *Ph.D. Thesis*, University of Cambridge, (1973).

Details the implementation of the CAP system at the University of Cambridge. Separate segments for data and C-lists. PSCS = per process capability list. MSL = master segment list which all capabilities indirect to. Hierarchy of subprocesses, each with its own PSCS which indirects to the parent's PSCS. PB = process base which contains enough information to restore the protection state. Problem with restarting subprocesses whose ancestors were interrupted (don't know whether returned to from below or resumed from above). Four distinguished registers: PDCS = indirectory for incarnation capabilities, I-indirectory = workspace local capabilities, G-indirectory = global capabilities, A-indirectory = argument capabilities. Lost object problem solved with garbage collection and reference counters. Problems listed in Chapter 6.

Walter, K. G., et. al.
Structured Specification of a Security Kernel, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 285-293.

The objective is to construct a reasonable model of the MULTICS system and to give an approach toward proving the model does not violate any security rules. Starts with the government's definition of security (M0) and embodies the rules in four axioms. The next model (M1) is more MULTICS specific with more axioms needed (more complex but closer to the real MULTICS). It is shown that all operations in M1 are secure since they satisfy the M0 axioms. An even more complex model, M2, is defined and built on M1. The authors claim this process can continue to yield models

which are closer and closer to the real MULTICS. By breaking the proof into steps, it is much easier than trying to do the entire proof in one step.

Ware, W. H.
Security and Privacy in Computer Systems, *Proceedings of the Spring Joint Computer Conference*, 30 (1967), pp. 279-282.

Ware, W. H.
Security and Privacy: Similarities and Differences, *Proceedings of the Spring Joint Computer Conference*, 30 (1967), pp. 287-290.

Weinstock, Charles B.
A Survey of Protection Systems, *Carnegie-Mellon University*, (July 1973).

Presents a summary of the important protection ideas prior to 1973. The work of Dennis and VanHorn, Graham, Lampson, Wulf and Morris are emphasized. Good review material.

Weissman, Clark
Secure Computer Operation With Virtual Machine Partitioning, *Proceedings of the National Computer Conference*, 44 (1975), pp. 929-934.

Reviews the steps used currently by the military when programs of different classification levels are to be executed on the same machine (periods processing with complete flush in between). Advocates using a VM architecture to allow different classification levels to execute concurrently. Cites IRIA paper authored with Belady as evidence that VM/370 is securable. Does not deal with possible covert channels that could be used to leak information from one partition to another. By keeping user interface with the VM monitor small, may be able to secure the VM.

Weissman, C.
Security Controls in the ADEPT-50 Time-Sharing System, *Proceedings of the Fall Joint Computer Conference*, 35 (1969), pp. 119-133.

Weissman, C.
Trade-Off Considerations in Security System Design, *System Development Corporation Report SP-3548*, (September 1970).

Wilkes, M. V.
Time-Sharing Computer Systems, Third Edition, American Elsevier, New York, (1975).

Wilner, W. T.
Burroughs B1700 Memory Utilization, *Proceedings of the Fall Joint Computer Conference*, 41 (1972), pp. 579-586.

Wilner, W. T.
Design of the Burroughs B1700, *Proceedings of the Fall Joint Computer Conference*, 41 (1972), pp. 489-497.

Wulf, W., et. al.
HYDRA: The Kernel of a Multiprocessing Operating System, *Communications of the ACM*, 17:6 (June 1974), pp. 337-345.

Implements a basic set of primitives from which operating systems can be built. A LNS is an object which is created on every procedure invocation. Its capability part contains incarnation permanent and temporary capabilities, its data part contains local data. Capabilities and data can both appear in an object. Can 'walk' from one object to another thru capabilities. Capabilities only manipulated by the kernel. Example of a bibliography protection problem.

Wulf, W. and R. Levin and C. Pierson
Overview of the HYDRA Operating System Development, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 122-131.

Briefly describes the hardware and address mapping in the HYDRA system. Advocates writing systems using structured programming and higher level languages. Uses functional decomposition as advocated by parnas but did not use the Parnas specifications due to recursive structures, evolution and context conditions. Gives

productivity figures of 14 instructions/day after adjusting for higher level language expansion as compared to 5-8 normally. Attributed to modular design.

Wulf, William A.
The HYDRA Operating System, Draft of a monograph on the HYDRA system, (1975).

This is the original draft from which the HYDRA session at SOSP was taken. Contains detailed information on the operation of HYDRA. Not for distribution.

DATA BASE MANAGEMENT

Allman, Eric and Michael Stonebraker and Gerald Held
Embedding a Relational Data Sublanguage in a General Purpose Programming Language, Proceedings of Conference on Data: Abstraction, Definition and Structure, *SIGPLAN Notices*, 11:4 (April 1976), pp. 25-35.

Discusses embedding the data base manipulation language, QUEL, into the general purpose "C" programming language on UNIX. Implemented by preprocessing the source program and substituting subroutine calls to processes connected by a pipe in place of the QUEL statements. Permitting data base definitions and data base command parameters to be defined at run time forces the parsing of statements at execution time. Recursion is not implemented due to language and space problems. Since types are not extensible in C, can perform type conversion at run time (number of types is enumerable). Local and global scope variables are mixed in some expressions.

Boyce, Raymond F. and Donald D. Chamberlin and W. Frank King III
Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage, *Communications of the ACM*, 18:11 (November 1975), pp. 621-628.

Excellent summary of the key components of the SQUARE language. The search for entries is done column-wise matching on a given attribute. The returned columns of the selected rows can be specified. Permits matching on any of the specified attributes (or) or on all of the attributes (and). Built in functions (count, max, min, ave, etc.) provide mathematical support. Free variables used to relate across data bases (dummy variables).

Chamberlin, D. D. and J. N. Gray and I. L. Traiger
Views, Authorization, and Locking in a Relational Data Base System, *Proceedings of the National Computer Conference*, 44 (1975), pp. 425-430.

There is a striking resemblence between the authorization primitives used and a capability system. Authorization checked at compile time. Advocates locking logical subsets of the data base at one time and not the entire data base.

Codd, E. F.
A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, 13:6 (June 1970), pp. 377-387.

Advocates independence of applications programs and data representation. Defines the normal form of a relation as one in which each element of the tuple is simple (i.e. not composed of another relation). Defines the operations of permutation, projection, join, composition and restriction on relations. No specific language implementation is discussed.

Codd, E. F.
"Further Normalization of the Data Base Relational Model," *Data Base Systems*, Courant Computer Science Symposium 6, Randall Rustin, Editor, Prentice-Hall, Englewood Cliffs, New Jersey, (1972), pp. 33-64.

Second and third normal forms are introduced to constrain the data base relations in such a way as to make the representation more independent of the queries. First normal form requires that no domain of a relation be a relation. Second normal form requires each non-prime attribute of a relation to be fully dependent on each candidate key. Third normal form eliminates transitive dependence of non-prime attributes on candidate keys. The number of admissible states (amount of information) in third normal form > second normal form > first normal form. Construction of the third normal form and second normal form from the first normal form is done by projecting a relation to form two or more other relations so no information is lost. The original relation is the join of the derived relations.

Date, C. J.
An Introduction to Database Systems, Addison-Wesley Publishing Company, (1975).

Date, C. J.
Relational Data Base Concepts, *Datamation*, 22:4 (April 1976), pp. 50-53.

Provides simple explanations for terms associated with relational data bases. The select operator selects rows from the relation matrix, the join operator concatenates two data bases based on one column, and the project operator selects columns from the relation matrix. Good introductory material.

King, Paul F. and Arthur J. Collmeyer
Database Sharing - An Efficient Mechanism for Supporting Concurrent Processes, *Proceedings of the National Computer Conference*, 42 (1973), pp. 271-275.

A lock-unlock mechanism is explained which allows concurrent readers and writers to access the same data base. Each writer requests that the records to be modified are locked (incremental record locking). A deadlock detection algorithm is given. To aid in deadlock recovery, a copy of each record reserved is given to the writer. When deadlock occurs and a writer must be aborted, the original data base will still be consistent. The data base is updated from the copy when the records are unlocked normally.

Olle, T. William
Current and Future Trends in Data Base Management Systems, *Proceedings of IFIP 1974*, (1974).

Reviews major design requirements of a DBMS such as data independence and interface design. Touches on the various types of data base design. Has a good bibliography at the end of the paper.

Owens, Richard C., Jr.
Primary Access Control in Large-Scale Time-Shared Decision Systems, *Project MAC Report MAC-TR-89, Cambridge, Massachusetts*, (July 1971).

Discusses the Project MAC Advanced Interactive Management System. Each data item read in is assigned a unique id. The items are grouped into Relational Data Sets. Access to any field in the RDS is based on the id of the requesting user. Nested accesses permitted are no access, manipulate with set theory operators, statistical access and print. Can specify that a user can only extract the mean, variance, etc. from a field. Can limit access based on the content of a field. A major problem is that information can be disclosed to unauthorized persons if certain operations are performed in a certain order in some cases (the problem of truth). Thus the method of deriving data must be checked.

Shoshani, A. and A. J. Bernstein
Synchronization in a Parallel- Accessed Data Base, *Communications of the ACM*, 12:11 (November 1969), pp. 604-607.

Identifies two problems with parallel accessed data bases. The first is that one primitive may be reading a record while another primitive is modifying it. The second is that one primitive may read a pointer to a record in the data base just before the record is destroyed and the space released. Gives a solution to deadlock with a set of assumptions.

Stonebraker, Michael and Eugene Wong
Access Control in a Relational Data Base Management System by Query Modification, *Proceedings of the ACM National Conference*, San Diego, California, (1974), pp. 180-186.

Protection is enforced by adding qualifications to the source statements submitted by the users. If the user is not aware of the restrictions placed on him, he could get a different subset than desired. Allows statistical (unprotected) access to the database. Claims efficiency advantages over embedding the protection mechanism in the accessing code. Might have space advantages over other methods. Only have replace, delete, combine and retrieve statements.

Zloof, Moshe M.
Query By Example, *Proceedings of the National Computer Conference*, 44 (1975), pp. 431-437.

Explains the query by example method by using numerous examples of increasing complexity. The main idea is that the data base can be arranged as a table and that relations between different data bases can be represented by dummy tokens in different columns.

INFORMATION THEORY

Ash, Robert
Information Theory, Interscience Publishers, John Wiley and Sons, (1965).

Cheng, M. C.
On the Computation of Capacity of Discrete Memoryless Channel, *Information and Control*, 24 (1974), pp. 292-298.

Gives a criteria for selecting t independent rows of the channel matrix for computing the channel capacity where t is the rank of the matrix.

Blackwell, David and Leo Breiman and A. J. Thomasian
The Capacities of Certain Channel Classes Under Random Coding, *Annals of Mathematical Statistics*, 31:3 (September 1960), pp. 558-567.

Analyzes three classes of channels: (1) fixed unknown channel, (2) arbitrarily varying channel, (3)channel selected by a jammer with knowledge of past inputs and outputs. Makes a case for random codes as opposed to pure codes.

Blackwell, David and Leo Breiman and A. J. Thomasian
The Capacity of a Class of Channels, *Annals of Mathematical Statistics*, 30:4 (December 1959), pp. 1229-1241.

Investigates the case where each channel input letter is transmitted over an unknown channel of a class of channels. The capacity of the class of channels is shown to be the sup over Q(x) of the inf over the channels comprising the class of channels, of I(X;Y). This is the foundation for a later paper on classes of channels.

Gallager, Robert G.
Information Theory and Reliable Communication, John Wiley and Sons, (1968).

Kotz, Samuel
Recent Results in Information Theory, *Journal of Applied Probability*, 3:1 (June 1966), pp. 1-93.

Excellent summary of results in information theory up to 1966. An extensive bibliography is included.

Shannon, Claude E.
The Zero Error Capacity of a Noisy Channel, *IRE Transactions on Information Theory*, IT-2 (1956), pp. 8-19.

The rate at which information can be transmitted over a finite discrete memoryless channel with zero decoding error is investigated. If all input letters are adjacent (both input letters can cause the same output letter with non-zero probability) then the zero error capacity is zero. Taking all combinations of non-adjacent letters with block length N will not always give the zero error capacity (dues give a lower bound). Two channels with the same adjacency matrix have the same zero error capacity. Gives bounds for the zero error capacity. An adjacency mapping method is used to determine the zero error capacity for most channels. Formulas for sum and product channels given. The zero error capacity may be increased by a noiseless feedback link.

Wolfowitz, J.
Coding Theorems of Information Theory, Second Edition, Springer-Verlag, (1964).

NETWORKING

Kleinrock, Leonard and William E. Naylor and Holger Opderbeck
A Study of Line Overhead in the Arpanet, *Communications of the ACM*, 19:1 (January 1976), pp. 3-13.

Classifies line traffic into control between adjacent IMPs, control between source IMP and destination IMP, control between hosts, background traffic and data. Measurements taken in May 1974. Most control messages (88.77 percent) were RFNM (request for next message. Two important parameters for line efficiency are message length and number of bits allocated per all request. About 0.59 of one percent of the line capacity was used for data. Overall, only 6.73 percent of the line capacity was used. Thus had much spare capacity. With saturated network, only 20 percent of line capacity could be used for data. Evaluated a proposed protocol by Cerf, Dalal and Sunshine that could possibly boost data usage to over 40 percent of line capacity.

OPERATING SYSTEM DESIGN

Dijkstra, E. W.
The Structure of THE Multiprogramming System, *Communications of the ACM*, 11:5 (May 1968), pp. 341-346.

Habermann, A. N. and Lawrence Flon and Lee Cooprider
Modularization and Hierarchy in a Family of Operating Systems, *Communications of the ACM*, 19:5 (May 1976), pp. 266-272.

Advocates building a family of operating systems in a hierarchical way where each system uses levels in common with other systems. Each level is defined functionally rather than by processes so many levels could be implemented by macros. Functional hierarchy checked at compile time and module boundaries (addressing) at run time. A virtual memory implementation is given in some detail.

Parnas, D. L.
A Technique for Software Module Specification With Examples, *Communications of the ACM*, 15:5 (May 1972), pp. 330-336.

Parnas, D. L.
On the Criteria to be Used in Decomposing Systems Into Modules, *Communications of the ACM*, 15:12 (December 1972), pp. 1053-1058.

PERFORMANCE EVALUATION

Batson, A., et. al.
Measurements of Segment Size, *Communications of the ACM*, 13:3 (March 1970), pp. 155-159.

Measurements were taken on a B5500 in a university environment running 90 percent ALGOL jobs on the average segment size (exclusive of the resident system) used. Results biased by compiler which allocates each row of an array to a different segment. Found the mean segment size for in-use segments to be 50 words and for user programs and data to be 24 words.

Bryan, G. E.
JOSS: 20,000 Hours at a Console - A Statistical Summary, *Proceedings of the Fall Joint Computer Conference*, 31 (1967), pp. 769-777.

Describes the RAND JOSS system completed in 1966 on the PDP-6. A drum is used for short term storage and a disk for long term storage. Disk traffic is light. The JOSS interpretor is 20 to 50 times slower than compiled code. Space on the disk is preallocated among users. Gives histograms on program size, compute time per task and per session, turn-around time, interaction time, and the average job characteristics. Since JOSS is not a general purpose time sharing system, the statistics are not applicable for those systems.

Buzen, Jeffrey
Analysis of System Bottlenecks Using a Queueing Network Model, *Proceedings of the ACM-SIGOPS Workshop on System Performance Evaluation*, Harvard University, (April 1971), pp. 82-103.

Develops a closed queueing model where all servers have exponentially distributed service times. The limitations of the model include a constant degree of multiprogramming, no processor overlap and the system is assumed to be at full load. Applies Gordon and Newell's method for the steady state solution of the closed model. Derives the probability distribution for the queue lengths and the processor utilization. Determines optimal performance obtained when the fastest processor has the longest expected queue.

Jalics, P. J. and W. C. Lynch
Selected Measurements of the PDP-10 TOPS-10 Timesharing Operating System, *Proceedings of IFIP 1974*, (1974).

Performed measurements on the TOPS-10 system and found that the system performance could possibly be enhanced by predicting what the processes would do next. Results include that CPU time used between I/O requests is hyperexponential with most less than 6 ms. that the next I/O device used is usually the same as the last one, that no disk head movement is needed 65 percent of the time, and that most files used sequentially. No particular relation between the software module that is running (e.g. compiler, editor, etc.) and the compute time between I/O. Was able to conduct meaningful measurements by saving histograms rather than saving every event. Gives some system performance measurements.

Schroeder, M. D.
Performance of the GE-645 Associative Memory While MULTICS is in Operation, *Proceedings of the ACM-SIGOPS Workshop on System Performance Evaluation*, Harvard University, (April 1971), pp. 227-245.

Describes in detail the use of the associative memory in the MULTICS address translation mechanism. An experiment which varied the number of associative registers that were actually used is described along with the attendant performance figures. For 16 registers, only an average of 1.03 core references were made (1 is the minimum possible) rather than the 4.0 that are necessary without associative registers. This result is a commentary on program locality in MULTICS programs as well as on the associative registers.

Sencer, M. A. and C. L. Sheng
An Analysis of Multiprogrammed Time-Sharing Computer Systems, *Proceedings of the National Computer Conference*, 42 (1973), pp. 87-91.

Defines a time-sharing system model where there are m groups of I/O processors (each group may have a different number of processors) and one group of CPU's. The model is analyzed for time in system and for congestion. The results of Jackson are used here. Each group of processors is treated as a M/M/infinity queue.

PROGRAMMING LANGUAGES

Allen, F. E. and J. Cocke
A Program Data Flow Analysis Procedure, *Communications of the ACM*, 19:3 (March 1976), pp. 137-147.

A summary of the interval analysis approach to static program analysis is given. The algorithm given determines what data definitions reach each node in the graph, what data items are upwards exposed from each node and its successors, what definitions are live on each edge of the graph.

Alsberg, Peter A.
Extendable Data Features in the Operating System Language OSL/2, *Proceedings of the Third Symposium on Operating Systems Principles*, Stanford University, (October 1971), pp. 31-34.

Each extended data type has two distinguished operations - fetch and store. If a variable of an extended type appears on the right hand side of an expression, fetch is called. If it appears on the left hand side, store is called. Appears to be a rather inflexible extension scheme. Some objects need more functions and with more mnemonic significance.

Balzer, R. M.
Dataless Programming, *Proceedings of the Fall Joint Computer Conference*, 31 (1967), pp. 535-544.

Advocates a language in which the data representation is divorced from the source program. The allowable operations on the data are abstract operations such as insert region, delete region, add region, etc. Sequencing thru the data structure done by either selecting all items that satisfy a certain condition or by marking the first record that satisfies the condition. In the latter case, a statement is provided to advance the pointer to the next record. User defined data structures are allowed if four accessing routines are defined. Cumbersome to use data which may be accessed as two types (character and integer for example). Also the four distinguished operations may not be the correct ones for all data structures. The data representation of a program may be changed without changing the source program. Can do same thing with proper modularity.

Berry, D. M.
Block Structure: Retention or Deletion, *Proceedings of the Third ACM Symposium on the Theory of Computing*, (May 1971).

Berry, D. M.
Introduction to OREGANO, Proceedings of the Symposium on Data Structures in Programming Languages, *SIGPLAN Notices*., 6:2 (February 1971), pp. 171-190.

Permits retention of blocks as long as the block can be referenced by an active processor (determined by reference counters/garbage collection). New types can be defined in terms of old types. Coroutine, recursive coroutines, tasks and general data structures can be expressed.

Berry, D. M., et. al.
On the Time Required for Retention, Source Publication Unknown.

Dahl, Ole-Johan and Kristen Nygaard
SIMULA - An ALGOL-Based Simulation Language, *Communications of the ACM*, 9:9 (September 1966), pp. 671-678.

Describes SIMULA as an event simulation language and an early version of the class concept that was later incorporated into SIMULA67. ALGOL60 is a subset of SIMULA. Automatic event list manipulation is provided. Processes in SIMULA correspond to event routines which are invoked according to the order of events on the event list. List processing is supported by allowing each process to have a predecessor and a successor process. A set head is used as a list starter. Data local to one process can be referenced by other processes.

Dennis, Jack B. and David P. Misunas
A Preliminary Architecture for a Basic Data-Flow Processor, Proceedings of the 2nd Annual Symposium on Computer Architecture., *Computer Architecture News (ACM-SIGARCH)*, 3:4 (December 1974), pp. 126-132.

The processor memory consists of cells which contain the two operands, the function to be performed and the destination address. When all operands in a cell are present, the contents of the cell are transported to the functional units thru an arbitration network. Conditionals and boolean operations are implemented by adding node types to the basic operator node type. A two level memory consisting of a cache for the most active cells

and instruction memory is presented. A detailed implementation is given.

DeRemer, Franklin L.
Simple LR(K) Grammars, *Communications of the ACM*, 14:7 (July 1971), pp. 453-460.

Gives an example of how to parse LR(0) grammars as previously done by knuth. Extends this procedure to the parsing of LR(1) then LR(K) grammars. Inadequate states are caused by the finite state machine being unable to distinguish whether a reduction or a further parsing is to be done. K denotes how many symbols the parser can 'look ahead' to resolve the ambiguity.

Dijkstra, Edsger W.
Guarded Commands, Nondeterminacy and Formal Derivation of Programs, *Communications of the ACM*., 18:8 (August 1975), pp. 453-457.

Defines two statements which can lead to nondeterministic execution: the IF construct which is composed of one or more statement lists each preceded by a boolean expression and the DO construct which also contains one or more statement lists preceded by a boolean expression. For the IF construct, only one statement list is chosen for execution from those with true guards. For the DO construct, statement lists are executed until all guards are false. A precondition function, WP(S,R), outputs the weakest condition so that condition R is satisfied after executing the statements, S. Objective is to be able to define programs which always halt and always halt in a known state. An example of the IF and DO construct derivation for problems is given.

Earley, Jay
Toward an Understanding of Data Structures, *Communications of the ACM*, 14:10 (October 1971), pp. 617-627.

Presents a model of data structures as nodes, atoms connected to nodes and links from one node to another. Can model stacks, queues, complex numbers, etc. with this model. Defines operations which manipulate the structures. Implement the algorithm in a high level model, define the links, nodes and atoms in lower level models until have a representation which is compatible with the target machine. Drawbacks are that atoms should be able to be turned into data structures, that values of atoms can not be restricted (hard to represent a sorted list), that it is hard to represent a class of graphs where one member of the set can not generate all members through the allowable transformations.

Flon, L. and A. N. Habermann
Towards the Construction of Verifiable Software Systems, Proceedings of Conference on Data: Abstraction, Definition and Structure, *SIGPLAN Notices*, 11:4 (April 1976), pp. 141-148.

Introduces *path expressions* as a method for limiting the sequence of operations on an abstract type to a manageable subset of all possible sequences and for providing implicit synchronization for concurrent operations. As in CLU, the representation of an abstract type can only be manipulated by the operations provided with the type. For operation sequence limitation, the allowable sequence is listed with possibly a notation on some operators denoting which can be repeated indefinitely and denoting exclusive selection. For synchronization, the path expression is an invariant which must be satisfied before the operation can be executed. If the path expression is not satisfied, the requestor is put to sleep. Thus implicit synchronization occurs. Only one operation can be performed on one object at a time but the same operation can be performed on different objects concurrently.

Goodenough, John B.
Exception Handling: Issues and a Proposed Notation, *Communications of the ACM*, 18:12 (December 1975), pp. 683-696.

Defines the terms necessary for exception handling. Views exceptions as a way for the user of an operation to extend the operation's domain or range by allowing user to deal with failures, by indicating the significance of a result and by providing a means to monitor the operation. An 'implementation independent' notation for attaching exception handlers to language statements is presented. Wants to make exception handling checkable by the compiler.

Goodenough, John B.
Structured Exception Handling, *Proceedings of the Second ACM Symposium on Principles of Programming Languages*, Palo Alto, California, (January 1975), pp. 204-224.

Irons, Edgar T.
Experience With an Extensible Language, *Communications of the ACM*, 13:1 (January 1970), pp. 31-40.

Describes a compiled language (IMP) which has been in use since 1965. New operations are defined by rules that look like grammar rules. One side may be an expression and the other a name for example. An algorithm will attempt to fill in the syntatic catagories if omitted by the user. Can evaluate semantic expressions when the parse tree is built or during subsequent walks around the tree. Replace the node containing the extended operation with the operation definition. The problem with operand dependent operators is that there must be a different operator defined for each type pair (e.g. real-real, real-integer, etc.). Ambiguities arising from extended types should be caught at grammar definition time. IMP does not catch them at all.

Jones, Anita K. and Barbara H. Liskov
An Access Control Facility for Programming Languages, Second International Conference on Software Engineering, San Francisco, California, October 1976, to be published.

Connects programming languages which offer protected extended types to capability systems (most notably HYDRA). Outlines a language very similar to CLU in which all type checking is done at compile time. Defines two types of protected objects: object type and data structures. Variable have a declared type and a declared set of rights for objects. The variable binding can be changed (natural implementation of the binding is by capabilities. Amplification done automatically on module entry. Binding for data structures requires all constituent elements to have the same type and the same rights (rights of the structure may be different. The static mechanism is compared with HYDRA's dynamic mechanism.

Kosinski, Paul R.
A Data Flow Language for Operating Systems Programming, Proceedings of the ACM SIGPLAN-SIGOPS Interface Meeting on Programming Languages - Operating Systems, *SIGPLAN Notices*, 8:9 (September 1973), pp. 89-93.

DFPL is composed of primitives which are connected in graph form to represent a program. Each primitive is executed at a time which depends on the availability of its input data. Hence parallelism is easily expressed.

Leavenworth, B. M.
Syntax Macros and Extended Translation, *Communications of the ACM*, 9:11 (November 1966), pp. 790-793.

Defines two types of macros. Syntax macros (SMACROS) add new statements to the defined language which may use previously defined new statements (nested macros) for extensibility. Function macros are the normal type of macros which returns a result. The smacro definition contains a dummy statement representation with strings (parameters) defined between the new keywords. Conditional expansion may be done based on which parameters and keywords are missing.

Ledgard, Henry F. and Michael Marcotty
A Genealogy of Control Structures, *Communications of the ACM*, 18:11 (November 1975), pp. 629-639.

Classifies control structures. D-structures (for Dijkstra) include actions, compositions, if-then-else and while-do which are one-in and one-out. D′-structures add if-then, repeat-until and case statements to the allowable D-structures. BJ structures consist of actions, compositions, if-then- else and omega structures. Also have RE (repeat-exit), REC (repeat-exit-cycle), DRE and DREC. Five levels of conversion from one structure to another are distinguished. Sequential composition, if-then-else and while-do are theoretically complete. Concludes that the need for more complex control structures above D or D′ is not proven and that the utility of the go to is questionable. Many references.

Liskov, Barbara and Stephen Zilles
An Approach to Abstraction, *Computation Structures Group Memo 88*, MIT Project MAC, (September 1973).

Abstraction by procedures not sufficient since abstract data structures as well as abstract operators must be defined. Abstraction hides the implementation details of an extended type. Type checking supplied. Abstract type definitions are declared in clusters. Extended operations have a compound name (type$operation) plus parameters. Users view types as indivisible. Abstract type variables change from indivisible to their component types when passed to a routine implementing that type. Nested clusters allow inner clusters to reference data structures of outer clusters. Contains a light review of previous extensible language efforts. The syntax of the language presented here differs in detail from later versions. Compiler would be empowered to insert the code for the extended operations in line for efficiency if necessary.

Liskov, Barbara
A Note on CLU, *Computation Structures Group Memo 112*, MIT Project MAC, (November 1974).

Decided that minor modifications to PL/1, PASCAL, EL1 or SIMULA67 would not implement definitions of operations for a type, the user not needing to know the storage structure of types, and permitting only defined operations to be used for a type. CLU is based on structured programming and modularity. Capability systems would be well suited for implementing CLU via domains for protecting the module user from the module writer and via protected enter capabilities.

Liskov, Barbara H.
Data Types and Program Correctness, *Proceedings of the National Computer Conference*, 44 (1975), pp. 285-286.

Emphasizes that the operations defined on an extended type are dependent on the type being defined. No set of 'distinguished operations' will suffice in every case. Encapsulating a data type definition with its operations will permit easier proofs that the implementation is correct. Outside of the type definition, the abstract properties of the type can be used and not the implementation. This could also improve program understandability.

Liskov, Barbara and Stephen Zilles
Programming With Abstract Data Types, Proceedings of a Symposium on Very High Level Languages, *SIGPLAN Notices*., 9:4 (April 1974), pp. 50-59.

Contains a CLU example in which data types can be defined at execution time rather than compile time. The descriptor unit appears to be able to provide the indirection mechanism necessary to permit updating clusters without recompiling all user programs. However, due to optimizing considerations, this is not done. Strong type checking done at execution time as in HYDRA.

Liskov, B. H. and S. N. Zilles
Specification Techniques for Data Abstractions, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 72-87.

Morris, James H., Jr
Protection in Programming Languages, *Communications of the ACM*, 16:1 (January 1973), pp. 15-21.

Advocates simulating the seal and unseal primatives with mark and trademark-list. The seal and unseal would be implemented by calling a special routine (get) which would construct (object,trademark-list) pairs. On presentation of a password, the unseal operation would be simulated and the original object returned. Note that trademark-list is a publically available function.

Morris, James H., Jr
Towards More Flexible Type Systems, *Proceedings of the Programming Symposium*, Paris, France, (April 1974), pp. 377-384.

The major reason for leaving loopholes in extended objects is so that a user can implement a function which is impossible or hard to do with the given primitives. For completeness a user should be able to generate all values and to discover all properties of a given value of an extended object. Proposes that the primitives be adequate to convert the new type to any other existing type. Polymorphic function must treat its parameters as opaque primitives.

Morris, James H., Jr
Tvpes Are Not Sets, *Proceedings of the ACM Symposium on the Principles of Programming Languages*, Boston, Massachusetts. (October 1973), pp. 120-124.

Two types of sealing operations. Transparent sealing allows the object to retain its previous properties and adds the sealed property (trademarks). Opaque sealing leaves only the sealed property with the object (users can not see the implementation). An upward coercion occurs when a more primitive type of object is manipulated directly like a more sophisticated object. A downward coercion occurs when a more sophisticated object is manipulated directly like a more primitive object. Advantages over SIMULA are: (1) can have strictly private module storage, (2) provides limited polymorphism (one subsystem will accept many types to perform its operation on).

McIlroy, M. Douglas
Macro Instruction Extensions of Compiler Languages, *Communications of the ACM*, 3:4 (April 1960), pp. 214-220.

Discusses several extensions to a macro processor to make macros more flexible. The extensions include nested macros, conditional macros, creating unique internal labels for macros, parenthesization for compound argument strings, and repetition of a macro from only one source invocation. This article seems to point to extensible languages as the next step although it was years before the idea was formalized.

Parnas, D. L. and John E. Shore and David Weiss
Abstract Types Defined as Classes of Variables, Proceedings of Conference on Data: Abstraction, Definition and Structure, *SIGPLAN Notices*, 11:4 (April 1976), pp. 149-154.

Presents a reasonable classification of previous type abstraction efforts and motivations for type extensions. Defines a type as an equivalence class of modes (e.g. integer, real) and thus permits the type operations to manipulate more than one mode. Defines spec-types as a type which can manipulate all modes for which the specifications can be satisfied. Defines rep-types as modes with the same representation on which useful common operators can be defined. Defines param-types as a type with modes generated by different settings of a parameter (e.g. array [M,N]). Defines variant types as having modes which have some common properties which can be manipulated. The motivations for this view of types stems from practical considerations rather than program proving considerations.

Reynolds, J. C.
GEDANKEN - A Simple Typeless Language Based on the Principle of Completeness and the Reference Concept, *Communications of the ACM*, 13:5 (May 1970), pp. 308-319.

Richards, Martin and Arthur Evans, Jr. and Robert F. Mabee
The BCPL Reference Manual, *MIT Project MAC Report MAC-TR-141*, Cambridge, Massachusetts, (December 1974).

Standish, Thomas A.
Extensibility in Programming Language Design, *Proceedings of the National Computer Conference*, 44 (1975), pp. 287-290.

Catagorizes extension techniques into paraphrase (define the new operation in terms of previously defined operations), orthophrase (add an orthogonal feature which will extend the space of attainable operations) and metaphrase (change interpretation rules of the language to give old statements new meanings). Hard extensions must be done by altering the internals of the language processor.

Wegbreit, Ben
The Treatment of Data Types in EL1, *Communications of the ACM*, 17:5 (May 1974), pp. 251-264.

Describes the EL1 programming language which is used in the ECL programming system at Harvard. The major feature of EL1 is that the user can define his own data abstractions, operator functions (for assignment, selection, printing and generation) and conversion routines for converting one type (or mode) to another. One limitation is that the user can not define an arbitrary set of operators but is constrained to the four above. Generic routines are used for operator definition and type conversion. Depending on type of abstraction used, the internal representation may or may not be hidden from the user. Any combination of compiled and interpreted routines may be executed. In summary, the type extension features seem overly complex. An approach as in CLU seems more reasonable even at a cost of not having the flexibility of EL1.

Wulf, W. A. and D. B. Russell and A. N. Habermann
BLISS: A Language for Systems Programming, *Communications of the ACM*, 14:12 (December 1971), pp. 780-790.

Queueing Theory

Allen, A. O.
Elements of Queueing Theory for System Design, *IBM Systems Journal*, 14:2 (1975), pp. 161-187.

This is an introductory paper for queueing theory which follows the cookbook approach. An extensive appendix of equations for the M/M/1 queue, the M/G/1 queue and the M/G/1 priority queue are given. The notion of percentile (e.g. the waiting time by which 90 percent of the customers are processed) is introduced. This is not discussed in Kleinrock. No queueing networks are discussed.

Anderson, H. A., Jr.
Approximating Pre-emptive Priority Dispatching in a Multiprogramming Model, *IBM Journal of Research and Development*, 17:11 (November 1973), pp. 533-539.

Contrasts the analysis of a computer system using different processor time and transition distributions (Posner's approach) with the usual single distribution approach. Processor sharing used for both the CPU and I/O devices. The state probabilities are derived. By ordering the required CPU time for each task type in an ascending manner, a pre-emptive discipline was approximated.

Babad, Jair m.
A Generalized Multi-Entrance Time-Sharing Priority Queue, *Journal of the ACM*, 22:2 (April 1975), pp. 231-247.

Solves a generalized version of the feedback scheduling algorithm by queueing theory. The scheduler is a mixture of feedback and priority scheduling where users with priority n enter at level n. Each level contains a separate queue for jobs in each priority class. When the job being serviced completes its quantum, it is placed in the queue containing jobs of its priority class at the next higher level. The job at the head of the highest priority queue in the lowest numbered non-empty level is chosen for execution. The expected response time and the expected waiting time in queue are derived.

Baskett, Forest, et. al.
Open, Closed, and Mixed Networks of Queues with Different Classes of Customers, *Journal of the ACM*, 22:2 (April 1975), pp. 248-260.

Describes queueing networks with four types of service centers: FCFS, processor sharing, infinite number of processors (no queueing) and preemptive LCFS. For all service centers but FCFS, each class of customers may have a distinct service time distribution. For FCFS, all classes of customers have the same service time distributions. Customers may change classes and service centers upon completion of service at one service center. The equilibrium state probabilities are solved for this type of queueing system.

Boyse, John W. and David R. Warn
A Straightforward Model for Computer Performance Prediction, *Computing Surveys*, 7:2 (June 1975), pp. 73-93.

Presents a seat-of-the-pants approach to modeling computer systems via queueing theory. Uses as input to the model data that would be readily available from most installations. To obtain data for possible extensions, the target system was executed to as closely as possible reflect the changes, then the results were scaled appropriately. The model analyzed was essentially a two service station model with the possibility for multiple CPU's.

Buzen, Jeffrey P.
Computational Algorithms for Closed Queueing Networks With Exponential Servers, *Communications of the ACM*, 16:9 (September 1973), pp. 527-531.

Explores computational algorithms which make the analysis of queueing networks easier. Both load dependent and load independent servers are treated. Only distribution of customers are considered.

Chandy, K. M. and U. Herzog and L. Woo
Approximate Analysis of General Queuing Networks, *IBM Journal of Research and Development*, 19:1 (January 1975), pp. 43-49.

Describes an iterative method for solving general queueing networks (including waiting time distribution) based on forming the complement of each queue in the network via Norton's theorem applied to queueing theory. The network is reduced to a two queue network which can be solved by using the method outlined in one of the references. The network to be analyzed is made to conform to local balance and is then analyzed.

Chandy, K. M. and U. Herzog and L. Woo
Parametric Analysis of Queuing Networks, *IBM Journal of Research and Development*, 19:1 (January 1975), pp. 36-42.

Argues that the rest of the queueing network outside of the subsystem of interest, can be modeled with an equivalent queue and server. This is analogous to Norton's circuit theorem. The theorem applies only to networks that satisfy local balance. The queue length distribution and the queue length distribution at arrival are shown to be equivalent for the original network and the simplified network. The service rate for the composit queue is set equal to the rate at which customers pass through the shorted subsystem of interest.

Chen, Peter Pin-Shan
Queueing Network Model of Interactive Computing Systems, *Proceedings of the IEEE*, 63:6 (June 1975), pp. 954-957.

Given some measurable parameters of a time-sharing system, the routing probabilities are determined. An approximation is made so that state-dependent routing is possible. A value is assumed for one parameter, the needed transition probabilities derived and the model solved by Buzen's method. Iterates until the assumed value and the solved value are close.

Gelenbe, Erol
On Approximate Computer System Models, *Journal of the ACM*, 22:2 (April 1975), pp. 261-269.

Discusses the diffusion approximation approach. A boundary condition is introduced which makes the model applicable to more lightly loaded situations. Prior models needed heavy traffic assumptions.

Gordon, William J. and Gordon F. Newell
Closed Queuing Systems With Exponential Servers, *Operations Research*, 15 (1967), pp. 254-265.

Jackson, James R.
Jobshop-Like Queueing Systems, *Management Science*, 10:1 (October 1963), pp. 131-142.

Jackson, James R.
Networks of Waiting Lines, *Operations Research*, 5 (1967), pp. 518-521.

Kleinrock, Leonard
Queueing Systems - Volume 1: Theory, John Wiley and Sons, (1975).

Kleinrock, Leonard
Queueing Systems - Volume 2: Computer Applications, John Wiley and Sons, (1976).

Muntz, Richard R.
Analytic Modeling of Interactive Systems, *Proceedings of the IEEE*, 63:6 (June 1975), pp. 946-953.

Gives an overview of the kinds of problems that can be solved with queueing theory. The author's belief is that future effort will be directed toward approximate solutions to general queueing networks rather than exact solutions to a relatively small set of queueing networks.

Muntz, R. R. and J. W. Wong
Efficient Computational Procedures for Closed Queueing Network Models, *Proceedings of the Seventh Hawaii International Conference on System Sciences*, (January 1974), pp. 33-36.

Presents an iterative procedure for calculating the normalization constant and the marginal probability that a specific service center is in a specified state. A similar method is given by Buzen but does not apply to the case where classes of jobs are allowed. This method should be used instead of the straightforward method for the calculations mentioned.

Posner, M. and B. Bernholtz
Closed Finite Queueing Networks With Time Lags and With Several Classes of Units, *Operations Research*, 16 (1968), pp. 977-985.

This paper solves queueing networks which have different classes of customers, FCFS single server service stations, delays between service centers and, most importantly, allow each class of customer to have its own exponential service rate. This method is

powerful enough to solve a CTSS-like system. The equilibrium distribution of the number in system can be found.

Reiser, M. and H. Kobayashi
Horner's Rule for the Evaluation of General Closed Queueing Networks, *Communications of the ACM*, 18:10 (October 1975), pp. 592-593.

This note gives an efficient algorithm for evaluating the product form solution G(M,N). Gives the formulas for the throughput and the mean queue size in terms of G(M,N).

Reiser, M. and H. Kobayashi
Queueing Networks With Multiple Closed Chains: Theory and Computational Algorithms, *IBM Journal of Research and Development*, 19:3 (May 1975), pp. 283-294.

Plows the same ground as Baskett, et. al. Provides for classes of customers with FCFS (all classes share the same distribution), ~~processor sharing, preemptive LCFS, and infinite server.~~ Uses generating functions in the analysis.

Scherr, Allan L.
An Analysis of Time-Shared Computer Systems, Research Monograph Number 36, The MIT press, Cambridge, Massachusetts, (1967).

This is a summary of the author's Ph.D. thesis which details his analysis of the CTSS system. Extensive measurements were taken of the system in operation and are compared with the analytic (Markov) model results and the simulation results. The interaction model for time-sharing systems is explained and used throughout the monograph.

Sevcik, Kenneth C.
Computer System Modelling and Analysis: Assessing Some Common Assumptions, *Proceedings of the Seventh Hawaii International Conference on System Sciences*, (January 1974), pp. 37-39.

Attacks the exponential distribution of job times because real jobs are closer to being hyperexponentially distributed, attacks zero preemption overhead as unrealistic (especially in choosing quantum lengths for round-robin scheduling), attacks id service times because of the presence of distinguishable job classes, attacks the independence of successive cpu times because of the existence of some correlation, and attacks the stationarity of distributions. Provides references for most of the arguments.

SCHEDULING

Agrawala, A. K. and R. M. Bryant
Models of Memory Scheduling, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 217-222.

Discusses swapping systems where more than one process can reside in memory at one time. The memory size is uniformly distributed between 1 and 100 blocks of memory (unrealistic). Considers FCFS and SMF (smallest memory size first). Uses four models, one with memory residence time, one with a processor-sharing scheduling algorithm and each of these is run with memory fragmentation allowed (no paging) and without (paging). Does not say why assumptions are a good model of computer systems. Uses simulation to obtain results. Forces an overload condition. For memory-residence FCFS with fragmentation observed rarely do adjacent regions free to coalesce. Also mean hole size slightly less than one-half the mean value of the largest hole size.

Bernstein, A. J.. and J. C. Sharp
A Policy-Driven Scheduler for a Time-Sharing System, *Communications of the ACM*, 14:2 (February 1971), pp. 74-78.

Proposes a scheduling algorithm which uses a policy function to determine how much service each process should receive by a certain process age. The processes which have received less service than promised are executed first. There is one policy function per class of processes. Discusses extending the concept to other resources besides the CPU. It is not clear how well this would work if the process does not continually demand service from the resource.

Browne, J. C. and Jean Lan and Forest Baskett
The Interaction of Multi-Programming Job Scheduling and CPU Scheduling, *Proceedings of the Fall Joint Computer Conference*, 41 (1972), pp. 13-21.

Modeled a CDC 6600 using the locally written operating systems UT-1 and UT-2. Job scheduler determines which jobs reside at the control points while CPU scheduling determines which control point receives the CPU. Simulator parameters based on measurements. Averages and standard deviations found by averaging ten runs rather than extending the length of one run. Job scheduling done by shortest time to run first (STF), smallest cost first (SCF), smallest memory first (SMF), FCFS. CPU scheduling done by RR (8 ms quantum), smallest time remaining (STR), shortest burst time next (SBT), longest burst time next (LBT). Thruput of FCFS worst which implies queueing models may not be accurate. SBT is theoretically the best. Results are that pre-emption needed for high thruput, that job scheduling affects thruput, RR most desirable if small process switching overhead and hyperexponential CPU times, that total I/O utilization dependent on both job and CPU scheduling, high thruput scheduling algorithms tend to have the most overhead, that best CPU utilization for RR or SBT, and that to maximize I/O utilization use SCF-RR or STF-RR.

Coffman, E. G., Jr. and I. Mitrani
Selecting a Scheduling Rule That Meets Pre-Specified Response Time Demands, *Proceedings of the Fifth Symposium on Operating Systems Principles*, The University of Texas at Austin, (November 1975), pp. 187-191.

Given a set of mean response times, one for each class of job, the arrival rate of each class, and the service rate of each class, a scheduling algorithm is found, if possible. The algorithm involves partitioning the CPU time into intervals where each interval has a different pre-emptive priority ordering.

Ellison, Carl M.
The Utah TENEX Scheduler, *Proceedings of the IEEE*, 63:6 (June 1975), pp. 940-945.

The original TENEX scheduler needed modification because the multi-level queue design could lock out a compute-bound user for up to fifteen minutes. Each user was given an income in 'Boyntons'/millisecond which was given only when a process was in execution. The scheduler charges a price for both CPU and memory pages used. If the price charged per millisecond is greater than the income, the POT for the process is decreased (a negative POT is ok). If more than a given percentage of a resource is purchased by processes in the black, the price is raised, and vice-versa. Could mean that a process is denied service after receiving partial service for a period of time if the price is very high and the machine is moderately loaded.

Kleinrock, Leonard
A Continuum of Time-Sharing Scheduling Algorithms, *Proceedings of the Spring Joint Computer Conference*, 36 (1970), pp. 453-458.

Discusses the class of scheduling algorithms which can be described by: (1) when a job enters the system it is assigned a priority of 0, (2) while the job waits in the queue its priority increases (or decreases) at rate alpha, (3) while the job is in service its priority increases (or decreases) at rate beta. Assumes the server can serve all jobs with the highest priority. Can get RR, FCFS, bulk service, LCFS, selfish RR (jobs stay in queue for a while until they start round robin, LCFS with seizure and LCFS with pickup. The selfish RR model is analyzed.

Lampson, B. W.
A Scheduling Philosophy for Multiprocessing Systems, *Communications of the ACM*, 11:5 (May 1968), pp. 347-360.

McKinney, J. M.
A Survey of Analytical Time-Sharing Models, *Computing Surveys*, 1:2 (June 1969), pp. 105-116.

Classifies scheduling algorithms into three catagories: round-robin, feedback, and external priorities. Since the execution times of jobs are generally not known in advance, the feedback (or FB_n) strategy is an attempt to implement the optimal shortest job first rule. Summarizes the research results in scheduling up to 1969. An annotated bibliography is given.

Pinkerton, Tad B.
Disk Scheduling: Theoretical vs. Practical Considerations, *Proceedings of the Seventh Hawaii International Conference on System Sciences*, (January 1974), pp. 64-66.

Discusses the measurement results of Lynch for MTS which showed that disk I/O requests to the same device were highly correlated. The two effects which tend to invalidate the uniform request probability models which have head seek time dominating the results are that a controller bottleneck may occur and that smaller but more numerous disk units tend to distribute the requests so there are fewer requests per unit.

Schrage, Linus
Optimal Scheduling Rules for a Single Processor, *Proceedings of the Seventh Hawaii International Conference on System Sciences*, (January 1974), pp. 31-32.

Tells the conditions under which FIFO, SPT/C (shortest processing time/cost), SRPT (shortest remaining processor time), FB (feedback), SIPT/C (shortest imminent processing time/cost), and SRPT/C (combination of SRPT and SPT/C) schedulers are optimal. Cost functions assumed linear.

Sherman, Stephen and Forest Baskett III and J. C. Browne
Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System, *Communications of the ACM*, 15:12 (December 1972), pp. 1063-1069.

Uses a trace-driven model of the UT-1 system as the basis for the measurements. The model was validated by comparing the simulator runs with the actual system measurements. Measurements showed CPU time distribution to be highly skewed (hyperexponential). Best algorithm (thruput time, CPU efficiency) was to choose job with shortest CPU time to next I/O request. Worst is to choose job with most CPU time to next I/O request. Preemptive algorithms necessary for better thruput and CPU efficiency. For predictive schedulers, need a bound on how long one job will compute so can not predict a service time larger than the bound. Round robin for skewed CPU distribution better than expected. Autocorrelations for CPU service times are small.

SOFTWARE TOOLS

Boehm, B. W. and R. K. McClean and D. B. Urfrig
Some Experience With Automated Aids To the Design of Large-Scale Reliable Software, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 105-113.

Believes that interface errors are more prevalent in large software projects than in small ones where one person can grasp the entire system being developed. Gives data that design errors more prevalent and cost more to fix than coding errors in large projects. Gives a list of error types and when they occur in the design. Explains the design assertion consistency checker which takes as input the input and output assertions for each module in the design and checks to determine whether all the interfaces are correct. For a 186 subsystem project, found seven incorrect interfaces.

Boyer, Robert S. and Bernard Elspas and Karl N. Levitt
SELECT -- A Formal System for Testing and Debugging Programs by Symbolic Execution, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 234-245.

Discusses a subsystem that analyzes paths thru a program written in a subset of LISP. Lists some reasons why proving programs is still unpractical. Symbolic execution of the program yields values for variables that are symbol strings. Loops are handled by counting each loop execution as a different path. The programmer can specify intermediate assertions and/or an output assertion. SELECT will try to generate test cases that violate the output condition. Function/subroutine calls are handled by macro-type expansion. On each branch, the branch indicated by the current data state is taken. If the alternative branch can be taken for some input data a backtrack point is established.

Brown, J. R. and R. H. Hoffman
Evaluating the Effectiveness of Software Verification - Practical Experience with an Automated Tool, *Proceedings of the Fall Joint Computer Conference*, 41 (1972), pp. 181-190.

Describes the flow program which determines the frequency with which program elements are exercised, determines the percentage of executable statements exercised, lists the subroutines not executed, determines the percentage of subroutines executed at least once, and total execution time in each subroutine. Case studies are given.

Reifer, Donald J.
Automated Aids for Reliable Software, Proceedings of the 1975 International Conference on Reliable Software, *SIGPLAN Notices*, 10:6 (June 1975), pp. 131-141.

Breaks down the software life cycle into three parts: conceptualization and requirements, development, and operations and maintenance. The most interesting part of the paper is the list of automated aids that could be available. Includes simulation, programming techniques, debugging aids and program support (like documentation). Points out that automated aids need to be integrated to provide better service. A good bibliography on software tools is included.

SYNCHRONIZATION

Dijkstra, E. W.
"Cooperating Sequential Processes," *Programming Languages*, F. Genuys, Editor, Academic Press, (1968), pp. 43-112.

Easton, W. B.
Process Synchronization Without Long-Term Interlock, *Proceedings of the Third ACM Symposium on Operating System Principles*, Stanford University, (October 1971), pp. 95-100.

Levitt, Karl N.
An Application of Program-Proving Techniques to the Verification of Synchronization Processes, *Proceedings of the Fall Joint Computer Conference*. 41 (1972), pp. 33-47.

Distribution List

Janel Chin	L-547
James Donnelley	L-307
William Frickel	L-307
Vik Hampel	L-307
Jeff Huskamp (5)	L-307
Steve Lai	L-307
Shig Tokubo	L-307
Doug Webb	L-307
Charles Wetherell	L-307
Mary Zosel	L-314
TID (15)	
TIC (27)	

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $2.25

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.25	426–450	11.75
126–150	5.50	451–475	12.00
151–175	6.00	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages: add $4.50 for each additional 100 page increment over 1,000 pages.