of a software error or hardware failure. Clearly, anything could happen if pointers and other items became available in ways not envisaged.

In the light of the argument that has been developed above, there are two alternatives open to us. One is to prevent the passage to a subordinate process of ENTER capabilities, or, what amounts to the same thing, to prevent them being used if they are so passed. If this is done, the writer of a subsystem has available to him the same protection mechanisms that were available to the writer of the main system. He can be given access to procedures that exist as part of the main system—that is to blocks of code—but he cannot be given access to protected procedures as such. He must construct for himself any protected procedures that he needs, using, if convenient, any already existing code to which he has access. This system is the one being adopted in the Cambridge CAP computer.

The alternative is to arrange that the information needed for restarting an interrupted process is communicated by the interrupt-servicing routine to the coordinator responsible for that process in such a form that it cannot be improperly used. The most straightforward way of implementing this idea would be for the interrupt-servicing routine to construct a protected procedure which, on entry, would restart the interrupted process. What would be passed to the coordinator responsible would be an ENTER capability for this procedure. If this were done, there would be no reason to restrict the passage of ENTER capabilities across boundaries of domains of coordination.

Although the passing of ENTER capabilities to a lower domain of coordination might be permissible, whether or not to do so would be entirely at the discretion of the user concerned and it is possible that he would make use of the facility sparingly. Problems might, in particular, be anticipated in the case of ENTER capabilities for protected procedures giving access to information that could not safely be accessed by more than one process at a time. This is the familiar situation in which some form of lock-out is conventionally employed. There is a danger that the information might become permanently locked out as a result of lack of attention to proper programming discipline on the part of a programmer operating in a lower domain of coordination.

References

Fabry, R. S. (1968). 'Preliminary description of a supervisor for a computer organised around capabilities'. *Quart. Prog. Rep.*, No. 18, Sect. IIA; Inst. Comp. Res., Chicago.

Needham, R. M. (1972). 'Protection systems and protection implementations', *AFIPS Conference Proceedings*, Vol. 41, p. 571.

Schroeder, M. D., and Saltzer, J. H. (1972). 'A hardware architecture for implementing protection rings', *CACM*, p. 157.

Wilkes, M. V. (1972). *Time-sharing computer systems.* Second Edition. Macdonald, London; American Elsevier, New York.

Book reviews

A SNOBOL 4 Primer, by R. E. Griswold and T. Madge. Griswold, 1973; 184 pages. (*Prentice-Hall International*)

Most readers will have heard of the string manipulation language SNOBOL; it is probably the most widely used general-purpose language for text handling, and is often used in teaching and research for such purposes as algebraic manipulation. Its most significant contribution to the development of programming is the introduction of abstract patterns as data types. The manual written by the inventors, and implementors, of the system is still the prime documentation for it. This is written in a clear, yet informal style, and gracefully includes a host of complicated additions to the language which have accumulated over the years.

The present book is intended as an introduction to SNOBOL for novices; one of the authors is one of the main designers of the language, so it is not surprising to find that its style and presentation are very similar to the previous manual. It is claimed that the book could form an introduction to computing for a complete beginner. In fact the introduction to the notion of an algorithm, and to the broader aspects of machines, such as storage, is extremely brief; in my opinion it would be baffling to a complete beginner, but would be suitable for students who have had a brief introduction to computing. The book gives a clear exposition of the main features of SNOBOL 4; it is, in fact, an exposition of a simplified and purified SNOBOL. It might be described as all the features that are safe use.

In summary the book could serve three roles: it is a good textbook, it is a good manual for most users, and it might serve as a basis for a standardised version of SNOBOL that does not rely on the original macro-implementation.

J. J. Florentin (London)

Communication Nets, by L. Kleinrock, 1973; 209 pages. (*Constable Publishers*, £1·50)

It is always somewhat difficult to review a book one has been recommending and using for many years. This book by L. Kleinrock comes in that category. When this book first came out, it was unique and essential reading for anyone who wished to analyse computer networks. The subject matter is well summarised by listing the chapters which are:

Chapter 1—Introduction
Chapter 2—Summary of results
Chapter 3—The problems of an exact mathematical solution to the general communication net
Chapter 4—Some new results from multiple channel systems
Chapter 5—Waiting times for certain queue disciplines
Chapter 6—Random routing procedures
Chapter 7—Simulation of communication nets
Chapter 8—Conclusion

The book is still very important in that it develops this subject in a consistent manner which makes it suitable as a textbook for third year undergraduate or graduate students. However, there has been considerable recent work on the subject and the book does not give an up-to-date rounded view in the same way as the recent book '*Computer Communications Networks*', in which Kleinrock also has an important article, which summarises many of his recent results.

The book is certainly an important item in a reference library, but it must be complemented by more recent publications, particularly those of Professor Kleinrock himself.

Peter T. Kirstein (London)