Book Reviews

RICHARD R. SHIVELY, Book Reviews Editor

In this section, the IEEE Computer Society publishes reviews of books in the computer field and related areas. Readers are invited to send comments on these reviews for possible publication in the Correspondence section of this TRANSACTIONS. Please address your comments and suggestions to the Book Reviews Editor: Richard R. Shively, Bell Telephone Laboratories, Whippany, N.J. 07981. The IEEE Computer Society does not necessarily endorse the opinions of the reviewers.


Since IBM's introduction of the System/360 and its operating system OS/360, probably no other aspect of that vast collection of programs and publications—the IBM OS Bibliography alone runs to 100 pages—has caused more heated discussions or been as poorly understood by its users as the job control language JCL. It is the one feature of the operating system that all users must employ; it is the gateway to the goodies of OS. JCL is not a programming language in the usual sense; it is not translated into machine instructions. Nor is it a command language, wherein each statement causes a specific action to take place. JCL is a job description language; it is a language used by the programmer to describe to the machine's operating system the work he has for it to do. To the System/360 operating system, a job is completely described by specifying a sequence of programs to be executed and, for each program, a list of external data sets required. Because there are so many possible combinations of programs and data, programs with all the various constraints applicable to them, data sets that can reside on any of a number of different devices and often in different ways on the same device, because of all these possibilities and because JCL avoids making assumptions about what the programmer intends, the job descriptions expressed in JCL often require considerable detail. This requirement for details that are often meaningless to the programmer is one of the factors which, as Gary DeWard Brown says in the Preface of System/360 Job Control Language, make JCL "... hard to learn and not always easy to use."

Another factor is the language itself; it is very low level as computer languages go. That is to say, it is not like any language used in disciplines other than computer science. One would expect that while each of the myriad special-purpose programming languages caters to its own community of users, the one computer language that they all must use would be as close as possible to the natural language that they all use, English. Instead, just the opposite is true. JCL is syntactically closest to the language used to write individual System/360 machine instructions. This assembly language is intimately related to the computer itself rather than to any application of the computer; so if it is the "work language" for anyone, it is the computer professional, not the accountant, physicist, or sociologist.

Still, with the millions invested in OS/360 programs, you can be sure that come hardware relocation or high water, OS will be around for a good while. So if, as Harry W. Cadow of IBM states in OS/360 Job Control Language, JCL is the manager of OS, then we had better try to become IBM computer "professionals," at least to the extent that we are conversant, if not fluent, in this language. Up to now, the only help we had in learning JCL was the official IBM manual.1 But that was like having a French dictionary when you are just learning French; the information is all there, you just do not understand the language the definitions are written in. What good is it to know that SPACE=(TRK, 60) means reserve 60 tracks of space on a direct-access storage device if you do not know what a track is or why 60 is better than 70 of whether you should be saying anything about space in the first place? Before getting down to such specifics as how to write certain parameters or which parameters to write or even what statements to write, a certain amount of knowledge is required about the computer hardware and about how the operating system handles jobs.

Messrs. Brown and Cadow both recognize this requirement, of course, and their books differ primarily in how much of this knowledge they assume the reader to have already. In his first chapter, Mr. Brown says his manual "... written for programmers just learning System/360 who have some familiarity with a higher level language but know little about JCL."

Chapter 2 is a rapid "Introduction to JCL and System/360." In 20 pages he presents the OS view of data processing from the bit on up and along the way defines most of the formidable OS vocabulary. The newcomer to the 360 will probably find it necessary—and desirable—to read this chapter more than once. In Chapter 3 Mr. Brown shows how specific JCL statements are used to describe a typical simple job; Chapter 4 summarizes JCL syntax. The next five chapters give detailed information on the most frequently used JCL statements. Of these, the data-definition (DD) statement is without doubt the most complicated and, consequently, the most troublesome. So it is not surprising that four complete chapters and most of a fifth are devoted to this statement, including a chapter on magnetic tapes and one on direct-access devices (disks and drums).

In somewhat of a departure, Mr. Brown has included a chapter on the linkage editor. This IBM-supplied program is not properly a part of the job control language, but it is ubiquitous in OS. It is the only user-invoked program that must be included in all OS installations, yet it is probably the least understood and most often misused program in the operating system. For this reason alone it belongs in an introduction to OS. Further, as Mr. Brown points out, it illustrates a number of JCL applications. Of special interest in this chapter are examples of how to create and use program libraries.

Most people who are acquainted with JCL find it difficult to speak of it in strictly factual terms. Mr. Brown, however, has done an admirable job. His book, which he properly calls a manual, will be a valuable reference source for the applications programmer who is looking for more understanding than you get from the IBM manual.

In OS/360 Job Control Language, Harry W. Cadow has used the System/360, OS, and JCL as vehicles for a textbook on the fundamentals of data processing, IBM style. In a sense his title is a misnomer since material on JCL occupies only about a third of the book. The first four chapters are a tutorial on System/360 hardware: central processor, storage, input/output channels, and devices. Chapter 5 sketches the possible role of a computer in a typical company. Then Chapters 6 and 7 discuss the operating system philosophy and organization in some detail. Chapter 6 covers programming conventions, the routing of a job through the system, and the supervision of work within a job. Chapter 7 is devoted to a review of I/O hardware and a discussion

---

1 IBM System/360 Operating System Job Control Language, Syst. Ref. Library (SRL) C28-4539-9. IBM also publishes Job Control Language Charts (C28-603-5), a coding aid that uses flow charts to help the programmer write specific JCL statements. With the nineteenth release of OS in June 1970, IBM split the JCL manual into a Reference Manual (C28-6704-0) and a Users Guide (C28-6703-1).
of how OS organizes and accesses data. The last four chapters address
JCL directly: three chapters to the three most common JCL statements
and one to catalogued procedures, a mechanism whereby frequently
used sets of JCL statements can be invoked with a single card.

Mr. Cadow's treatments of System/360 hardware and of the oper-
ating system are thorough and clear and his conversational style and
use of real-world examples put a little life into an otherwise dry subject.
As a matter of personal taste, I feel that the writing becomes at times
a little too slick—a posed picture of a pretty girl at a Model 40 shown,
as usual, flipping one of the console switches (something which, by
the way, is almost never done); at times a little too cute—"OS, pro-
nounced oz, as in 'the wizard of,'" and at times a little egocentric—
"the computer age has come to the designer's rescue; engineers have
designed an electronic sketch pad called the IBM 2250 . . ." or "the
operating system [as defined as] a collection of programs written and
supplied by IBM for a data-processing installation to use . . ." It
begins to sound like a sales release. Then again, it seems a bit presump-
tuous to present a description of the hardware and operating system of
one manufacturer—especially one's own company—as a text on "the
fundamentals of data processing," since there are, of course, computing
systems based on radically different machine organizations and operat-
ing system philosophies that are as much a part of the data processing
art as OS/360.

A choice between these two books would have to be based largely
on whether or not the buyer felt he needed the extensive background
material in Mr. Cadow's text. Both books supply background infor-
mation that goes well beyond current IBM manuals, but inexplicably
both authors have neglected any discussion of JCL diagnostic messages.
This is a serious omission since the novice can expect to encounter diag-
nostics frequently and since the messages generated by OS are con-
sistently vague and misleading.

If you are looking for a philosophers' stone to transmute JCL into
a simple, straightforward language, forget it; it cannot be done. But if
what you need is a good explanation of the philosophy and language
of IBM's OS/360 and a guide to the intelligent use of JCL—and you
probably do if you use the 360—then Mr. Cadow's detailed tutorial for
the newcomer to OS or Mr. Brown's more concise manual for the ini-
tiate will fill the bill.

STEPHEN CAMPBELL
Bell Telephone Lab., Inc.
Whippany, N. J. 07981

B72-2 A Compiler Generator—William C. McKeeman, James J.
Horning, and David B. Wortman (Englewood Cliffs, N.J.: Prentice-
Hall, 1970, 527 pp., $15.00).

The authors' compiler generator is a translator-writing system
(TWS) based on their derivative of PL/I called XPL. Of course, XPL
is designed to be self-compiling. The book can be viewed either as a
User's Manual for the XPL system or as an introduction to the con-
tuction of TWS's. These dual approaches complement each other since
one provides the illustration of the theoretical principles studied in the
other.

This practical approach of combining a text on compiler con-
struction with the documentation of their TWS is reflected in the authors'organization of the book. The second half of the 500 pages
is devoted to appendices containing XPL listings of the various parts
of their TWS: XCOM, ANALYSER, etc. These listings are well com-
mented and provide excellent detailed documentation of the XPL-TWS.
The first half of the book is evenly split into two parts called "Theory"
and "Practice." "Theory" provides the background for understanding
the illustration given in "Practice," namely, the description of the im-
plementation of the XPL-TWS. The preparation for practice by pro-
viding good theoretical background extends to the chapter level within
the two parts. Each chapter in "Theory" provides the background for the
respective chapter in "Practise." Thus the first "Theory" chapter,
which describes translators in general, prepares for the first "Practice"
chapter, which describes the XPL-TWS. The second "Theory" chapter,
the description of languages (BNF), prepares for the second "Practice"
chapter, programming in BNF, including the use of Analyser, a pro-
gram that determines whether a BNF grammar is acceptable for the
parsig algorithm used in this TWS. The third and fourth "Theory"
chapters, on translation (trees, Polish notation, scope, and nested
scopes) and parsing algorithms (for LR(k) grammars), prepare for the
third and fourth "Practice" chapters on XCOM, the translator of this
TWS, and on SKELET0N—a simplified version of XCOM. The final
"Theory" chapter, on constructing parsing decision tables, prepares for
the final "Practice" chapter, on Analyser and its construction.

The authors' clear presentation and ready wit contribute greatly
to this book's readability. These are most obvious in many quotes
sprinkled throughout. For instance, introducing the chapter, "The
Description of Languages," is the quote from Through the Looking
Glass, by Lewis Carroll:

... The name of the song is called 'Haddocks' Eyes!'”
"Oh, that's the name of the song, is it?" Alice said trying to feel
interested.

"No, you don't understand," the Knight said, looking a little
vexed. "That's what the name is called. The name really is, The
Aged Aged Man."

“I ought to have said 'That's what the song is called?'” Alice
corrected herself.

"No, you oughtn't; that's quite another thing! The song is called
Ways and Means: but that's only what it is called you know!"

"Well, what is the song then?" said Alice, who was by this time
completely bewildered.

"I was coming to that," the Knight said. "The song really is 'A-
Sitting on a Gate': and the tune's my own invention.

To introduce students to compiler construction without letting them
practice with some TWS should be a crime. This book, which contains
theoretical background, a User's Manual for a TWS, and listings of the
TWS, should help instructors avoid criminal charges.

THOMAS F. ANDERSON
Bell Telephone Lab., Inc.
Whippany, N. J. 07981

B72-3 Computer Structures, Readings and Examples—C. Gordon Bell

This book represents a major contribution to the study of the archi-
tecture of computer systems. It will be, I believe, must reading for any
serious student of the field.

The book consists of introductory material (approximately 100
pages), which includes the definition of two quite imaginative languages
for representing computer systems and the computing process. The re-
mainder of the book consists of a collection of papers describing almost
40 classic computer organizations. Accompanying these organizations
are, where appropriate, supplementary detailed descriptions of the
processors using their notation.

The authors introduce the notion that a complete processor de-
scription can be accomplished only by a hierarchy of languages; thus
the language of the physicist in describing a device is different from the
language of the circuit designer, which in turn is distinct from the
switching theorist's definition of logical functions. In any event, the
two major levels of this descriptive hierarchy that the authors adddress
are the processor-memory-switch level (PMS) and the instruction-set
processor (ISP) descriptive system. The PMS system is largely a short-
hand notation for describing a block diagram from the gross compo-
nents (physical boxes) of the system; thus the PMS description would
use the processor, memory, switch, and control unit as its basic com-
ponents. It seems to this reviewer to be a nice, simple, unified shorthand
for doing what most designers already do with block diagrammatic
descriptions of processors, memory, and I/O configurations. The notion
can be easily extended to include as much detail as necessary. Insofar
as it is graphic and familiar, it is easily used and more useful than
certain other attempts at formal descriptions of computer systems.

The ISP descriptive language is an attempt to define the processor