

6

Debugging

CHAPTER OUTLINE

Overview

Errors in Compilation

Common Compilation Errors

Errors in Execution

File Status Codes

Tips for Debugging

Cross-Reference Listing

DISPLAY Statement

Interactive Debugger

The Structured Walkthrough

Summary

Fill-in

True/False

Problems

OBJECTIVES

After reading this chapter you will be able to:

- Distinguish between errors in compilation and execution; correct typical compilation errors.
- Use the DISPLAY statement as a debugging tool.
- Explain how an interactive debugger can be used to find and correct execution errors.
- Describe the use of file status codes in correcting data management errors.
- Explain what is meant by a structured walkthrough; be able to participate as reviewer, reviewee, moderator, or secretary.

OVERVIEW

Very few computer programs run successfully on the first attempt. Indeed, the programmer is realistically expected to make errors, and an important test of a good programmer is not whether he or she makes mistakes, but how quickly he or she is able to detect and correct the errors. Since this process is such an integral part of programming, an entire chapter is devoted to debugging. We consider errors in both compilation and execution.

Compilation errors occur during the translation of COBOL to machine language and are caused by a mistake in COBOL syntax, for example, a missing period or an entry in a wrong column. Execution errors result after the program has been translated to machine language and produce results that are different from what the programmer expected or intended.

Compilation errors are easy to find because the compiler produces an explicit error message. Execution errors are more difficult to detect and may require the use of additional debugging tools, such as the insertion of DISPLAY statements into a program and/or the use of an interactive debugger. The chapter also considers the structured walkthrough as a means of reducing errors before they occur.

Errors in Compilation

Compilation is the process of translating a source (COBOL) program into machine language. Any mistake in COBOL syntax causes the compiler to make an assumption in the interpretation of the statement in which the error occurs, or, worse yet, makes it impossible for the compiler to interpret the statement at all. Either way a *compilation error* results.

Some errors are less severe than others; for example, the compiler is generally able to guess the programmer's intent when periods are omitted in the Data Division, whereas it is unable to decipher a misspelled reserved word. Accordingly, most compilers provide different levels of *compiler diagnostics* (error messages) according to the severity of the error. Micro Focus Personal COBOL for Windows, for example, produces five types of error messages, which are listed in order of increasing severity. Other compilers have similar classifications. Consider:

I Informational Diagnostics	Indicates a coding inefficiency or other condition (for example, an incompatibility with the ANS standard). The program will compile correctly.
W Warning Diagnostics	The statement is syntactically correct, but the source of a potential problem. A program can compile and execute with several W-level diagnostics present; however, ignoring these messages could lead to errors in execution.
E Error Diagnostics	The statement is incorrect as written, and requires the compiler to make an assumption in order to complete the compilation. You may wish to correct the program in case the compiler's assumption is not what you intended.
S Severe Diagnostics	A severe error in that the compiler cannot make corrections and therefore cannot generate object instructions. Any statement flagged as an S-level error is ignored and treated as if it were not present in the program.
U Unrecoverable Diagnostics	An error of such severity that the compiler does not know what to do and cannot continue. U-level diagnostics are extremely rare, and you practically have to submit a Visual BASIC program to the COBOL compiler to cause a U-level message.

The COBOL compiler tends to rub salt in a wound in the sense that an error in one statement can cause error messages in other statements that appear correct. For example, should you have an S-level error in a SELECT statement, the compiler will flag the error, ignore the SELECT statement, and then flag any other statements that reference that file even though those other statements are correct.

Often simple mistakes such as omitting a line or misspelling a reserved word can lead to a long and sometimes confusing set of error messages. The only consolation is that compiler errors can disappear as quickly as they occurred. Correction of the misspelled word or insertion of the missing statement will often eliminate several errors at once.

Proficiency in debugging comes from experience—the more programs you write, the better you become. You may correct the errors in the order they appear

(our preference), or in the order of severity (from Unrecoverable, Severe, Error, Warning, to Informational), or even haphazardly as you find them. Whichever way you choose, try to find the mistakes as quickly as possible and without wasting time. Moreover, don't spend too much time on any single error; instead, if you are stuck, skip the error temporarily and continue to the next, eliminating as many errors as you can before you recompile.

Figure 6.1 Tuition Billing Program with Compilation Errors

```

1  IDENTIFICATION DIVISION.
2  PROGRAM-ID.    TUIT6COM.
3  AUTHOR.       CAROL VAZQUEZ VILLAR.
4
5  ENVIRONMENT DIVISION.
6  INPUT-OUTPUT SECTION.
7  FILE-CONTROL.
8      SELECT STUDENT-FILE  ASSIGN TO 'A:\CHAPTR06\TUITION.DAT'
9          ORGANIZATION IS LINE SEQUENTIAL.
10     SELECT PRINT-FILE
11         ASSIGN TO PRINTER.
12
13  DATA DIVISION.
14  FILE SECTION.
15  FD STUDENT-FILE
16     RECORD CONTAINS 27 CHARACTERS.
17  01 STUDENT-RECORD.
18     05 STU-NAME.
19         10 STU-LAST-NAME    PIC X(15).
20         10 STU-INITIALS    PIC XX.
21     05 STU-CREDITS        PIC 9(2).
22     05 STU-UNION-MEMBER   PIC X.
23     05 STU-SCHOLARSHIP    PIC 9(4).
24     05 STU-GPA            PIC 9V99.
25
26  FD PRINT-FILE
27     RECORD CONTAINS 132 CHARACTERS.
28  01 PRINT-LINE          PIC X(132).
29
30  WORKING-STORAGE SECTION.
31  01 DATA-REMAINS-SWITCH  PIC X(2)  VALUE SPACES.
32
33  01 INDIVIDUAL-CALCULATIONS.
34     05 IND-TUITION        PIC 9(4)  VALUE ZEROS.
35     05 IND-ACTIVITY-FEE   PIC 9(2)  VALUE ZEROS.
36     05 IND-UNION-FEE      PIC 9(2)  VALUE ZEROS.
37     05 IND-SCHOLARSHIP    PIC 9(3)  VALUE ZEROS.
38     05 IND-BILL           PIC 9(6)  VALUE ZEROS.
39
40  01 UNIVERSITY-TOTALS.
41     05 UNI-TUITION        PIC 9(6)  VALUE ZEROS.
42     05 UNI UNION FEE      PIC 9(4)  VALUE ZEROS.

```

Hyphens missing

Figure 6.1 (continued)

```

43      05 UNI-ACTIVITY-FEE      PIC 9(4) VALUE ZEROS.
44      05 UNI-SCHOLARSHIP      PIC X(6) VALUE ZEROS.
45      05 UNI-IND-BILL         PIC 9(6) VALUE ZEROS.
46
47      01 CONSTANTS-AND-RATES.
48      05 PRICE-PER-CREDIT     PIC 9(3) VALUE 200.
49      05 UNION-FEE           PIC 9(2) VALUE 25.
50      05 ACTIVITY-FEES.
51          10 1ST-ACTIVITY-FEE PIC 99  VALUE 25.
52          10 1ST-CREDIT-LIMIT PIC 99  VALUE 6.
53          10 2ND-ACTIVITY-FEE PIC 99  VALUE 50.
54          10 2ND-CREDIT-LIMIT PIC 99  VALUE 12.
55          10 3RD-ACTIVITY-FEE PIC 99  VALUE 75.
56      05 MINIMUM-SCHOLAR-GPA PIC 9V9  VALUE 2.5.
57
58      01 HEADING-LINE.
59      05 FILLER                PIC X    VALUE SPACES.
60      05 FILLER                PIC X(12) VALUE 'STUDENT NAME'.
61      05 FILLER                PIC X(10) VALUE SPACES.
62      05 FILLER                PIC X(7)  VALUE 'CREDITS'.
63      05 FILLER                PIC X(2)  VALUE SPACES.
64      05 FILLER                PIC X(7)  VALUE 'TUITION'.
65      05 FILLER                PIC X(2)  VALUE SPACES.
66      05 FILLER                PIC X(9)  VALUE 'UNION FEE'.
67      05 FILLER                PIC X(2)  VALUE SPACES.
68      05 FILLER                PIC X(7)  VALUE 'ACT FEE'.
69      05 FILLER                PIC X(2)  VALUE SPACES.
70      05 FILLER                PIC X(11) VALUE 'SCHOLARSHIP'.
71      05 FILLER                PIC X(2)  VALUE SPACES.
72      05 FILLER                PIC X(10) VALUE 'TOTAL BILL'.
73      05 FILLER                PIC X(48) VALUE SPACES.
74
75      01 DETAIL-LINE.
76      05 FILLER                PIC X    VALUE SPACES.
77      05 DET-LAST-NAME         PIC X(15).
78      05 FILLER                PIC X(2)  VALUE SPACES.
79      05 DET-INITIALS         PIC X(2).
80      05 FILLER                PIC X(5)  VALUE SPACES.
81      05 STU-CREDITS           PIC 9(2).
82      05 FILLER                PIC X(6)  VALUE SPACES.
83      05 DET-TUITION          PIC 9(6).
84      05 FILLER                PIC X(7)  VALUE SPACES.
85      05 DET-UNION-FEE        PIC 9(3).
86      05 FILLER                PIC X(6)  VALUE SPACES.
87      05 DET-ACTIVITY-FEE     PIC 9(3).
88      05 FILLER                PIC X(8)  VALUE SPACES.
89      05 DET-SCHOLARSHIP      PIC 9(5).
90      05 FILLER                PIC X(6)  VALUE SPACES.
91      05 DET-IND-BILL         PIC 9(6).
92      05 FILLER                PIC X(49) VALUE SPACES.

```

Alphanumeric picture is not permitted for numeric calculation

Period missing

Should be DET-CREDITS

Figure 6.1 (continued)

```

93
94     01 DASH-LINE.
95         05 FILLER                PIC X(31) VALUE SPACES.
96         05 FILLER                PIC X(8)  VALUE ALL '-'.
97         05 FILLER                PIC X(2)  VALUE SPACES.
98         05 FILLER                PIC X(8)  VALUE ALL '-'.
99         05 FILLER                PIC X(2)  VALUE SPACES.
100        05 FILLER                PIC X(7)  VALUE ALL '-'.
101        05 FILLER                PIC X(6)  VALUE SPACES.
102        05 FILLER                PIC X(7)  VALUE ALL '-'.
103        05 FILLER                PIC X(5)  VALUE SPACES.
104        05 FILLER                PIC X(7)  VALUE ALL '-'.
105        05 FILLER                PIC X(49) VALUE SPACES.
106
107     01 TOTAL-LINE.
108         05 FILLER                PIC X(8)  VALUE SPACES.
109         05 FILLER                PIC X(17)
110             VALUE 'UNIVERSITY TOTALS'.
111         05 FILLER                PIC X(8)  VALUE SPACES.
112         05 TOT-TUITION           PIC 9(6) .
113         05 FILLER                PIC X(6)  VALUE SPACES.
114         05 TOT-UNION-FEE         PIC 9(4) .
115         05 FILLER                PIC X(5)  VALUE SPACES.
116         05 TOT-ACTIVITY-FEE     PIC 9(4) .
117         05 FILLER                PIC X(7)  VALUE SPACES.
118         05 TOT-SCHOLARSHIP      PIC 9(6) .
119         05 FILLER                PIC X(6)  VALUE SPACES.
120         05 TOT-IND-BILL         PIC 9(6) .
121         05 FILLER                PIC X(49) VALUE SPACES.
122
123     PROCEDURE DIVISION.
124     START. Reserved word used as paragraph name
125         OPEN INPUT STUDENT-FILE
126             OUTPUT PRINT-FILE.
127         PERFORM WRITE-HEADING-LINE.
128         PERFORM READ-STUDENT-FILE.
129         PERFORM PROCESS-STUDENT-RECORD
130             UNTIL DATA-REMAINS-SWITCH = 'NO'.
131         PERFORM WRITE-UNIVERSITY-TOTALS.
132         CLOSE STUDENT-FILE
133             PRINT-FILE.
134         STOP RUN.
135
136     WRITE-HEADING-LINE.
137         MOVE HEADING-LINE TO PRINT-LINE.
138         WRITE PRINT-LINE
139             AFTER ADVANCING PAGE.
140         MOVE SPACES TO PRINT-LINE.
141         WRITE PRINT-LINE.
142

```

Figure 6.1 (continued)

```

143 READ-STUDENT-FILE.
144 READ STUDNET-FILE
145     AT END MOVE 'NO' TO DATA-REMAINS-SWITCH
146 END-READ.
147
148 PROCESS-STUDENT-RECORD.
149     PERFORM COMPUTE-INDIVIDUAL-BILL.
150     PERFORM INCREMENT-UNIVERSITY-TOTALS
151     PERFORM WRITE-DETAIL-LINE.
152     PERFORM READ-STUDENT-FILE.
153
154 COMPUTE-INDIVIDUAL-BILL.
155     PERFORM COMPUTE-TUITION.
156     PERFORM COMPUTE-UNION-FEE.
157     PERFORM COMPUTE-ACTIVITY-FEE.
158     PERFORM COMPUTE-SCHOLARSHIP.
159     COMPUTE IND-BILL = IND-TUITION + IND-UNION-FEE + IND-ACTIVITY
160                     - IND-SCHOLARSHIP.
161
162 COMPUTE-TUITION.
163     COMPUTE IND-TUITION =PRICE-PER-CREDIT * STU-CREDITS.
164
165 COMPUTE-UNION-FEE.
166     IF STU-UNION-MEMBER = 'Y'
167         MOVE UNION-FEE TO IND-UNION-FEE
168     ELSE
169         MOVE ZERO TO IND-UNION-FEE
170     END-IF.
171
172 COMPUTE-ACTIVITY-FEE.
173     EVALUATE TRUE
174         WHEN STU-CREDITS <= 1ST-CREDIT-LIMIT
175             MOVE 1ST-ACTIVITY-FEE TO IND-ACTIVITY-FEE
176         WHEN STU-CREDITS > 1ST-CREDIT-LIMIT
177             AND STU-CREDITS <= 2ND-CREDIT-LIMIT
178             MOVE 2ND-ACTIVITY-FEE TO IND-ACTIVITY-FEE
179         WHEN STU-CREDITS > 2ND-CREDIT-LIMIT
180             MOVE 3RD-ACTIVITY-FEE TO IND-ACTIVITY-FEE
181         WHEN OTHER
182             DISPLAY 'INVALID CREDITS FOR: ' STU-NAME
183     END-EVALUATE.
184
185 COMPUTE-SCHOLARSHIP.
186     IF STU-GPA > MINIMUM-SCHOLAR-GPA
187         MOVE STU-SCHOLARSHIP TO IND-SCHOLARSHIP
188     ELSE
189         MOVE ZERO TO IND-SCHOLARSHIP
190     END-IF.
191
192 INCREMENT-UNIVERSITY-TOTALS.

```

Should be STUDENT-FILE

FEE extends past column 72

Missing space after =

Multiple definition in lines 21 and 81

Inconsistent PIC clauses in lines 23 and 37

Figure 6.1 (continued)

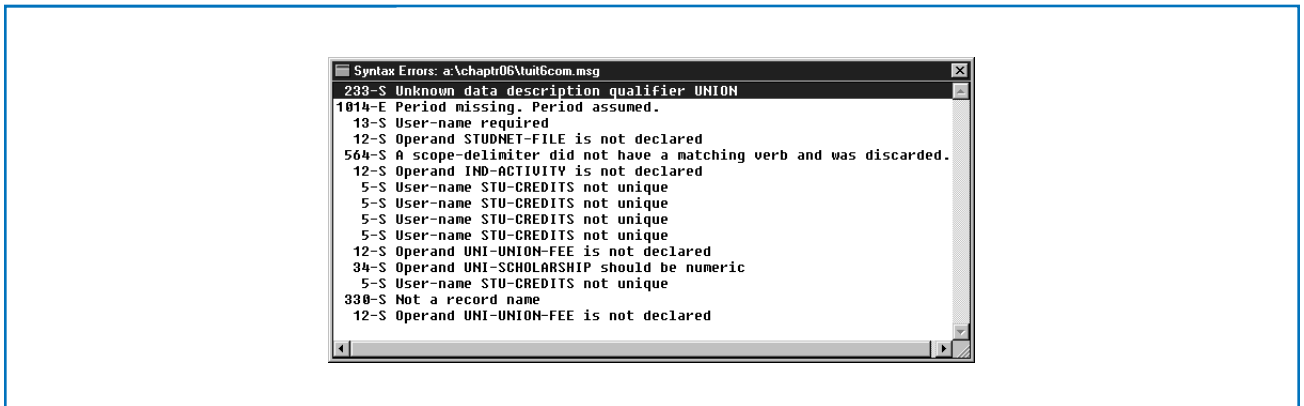
```

193      ADD IND-TUITION      TO UNI-TUITION.
194      ADD IND-UNION-FEE   TO UNI-UNION-FEE.
195      ADD IND-ACTIVITY-FEE TO UNI-ACTIVITY-FEE.
196      ADD IND-SCHOLARSHIP TO UNI-SCHOLARSHIP.
197      ADD IND-BILL        TO UNI-IND-BILL.
198
199  WRITE-DETAIL-LINE.
200      MOVE STU-LAST-NAME TO DET-LAST-NAME.
201      MOVE STU-INITIALS TO DET-INITIALS.
202      MOVE STU-CREDITS TO DET-CREDITS.
203      MOVE IND-TUITION TO DET-TUITION.
204      MOVE IND-UNION-FEE TO DET-UNION-FEE.
205      MOVE IND-ACTIVITY-FEE TO DET-ACTIVITY-FEE.
206      MOVE IND-SCHOLARSHIP TO DET-SCHOLARSHIP.
207      MOVE IND-BILL TO DET-IND-BILL.
208      MOVE DETAIL-LINE TO PRINT-LINE.
209      WRITE PRINT-FILE
210          AFTER ADVANCING 1 LINE.
211
212  WRITE-UNIVERSITY-TOTALS.
213      MOVE DASH-LINE TO PRINT-LINE.
214      WRITE PRINT-LINE.
215      MOVE UNI-TUITION TO TOT-TUITION.
216      MOVE UNI-UNION-FEE TO TOT-UNION-FEE.
217      MOVE UNI-ACTIVITY-FEE TO TOT-ACTIVITY-FEE.
218      MOVE UNI-SCHOLARSHIP TO TOT-SCHOLARSHIP.
219      MOVE UNI-IND-BILL TO TOT-IND-BILL.
220      MOVE TOTAL-LINE TO PRINT-LINE.
221      WRITE PRINT-LINE
222          AFTER ADVANCING 1 LINE.

```

Put hyphens in Working-Storage definition line 42
PIC X(6) used in definition
Multiple definitions in lines 21 and 81
Not defined in detail line
Should be PRINT-LINE
Put hyphens in Working-Storage definition line 42
PIC X(6) used in definition

To give you a better feel of what to expect from your programs, we have taken the Tuition Billing program from Chapter 5 and deliberately changed several of the statements to cause compilation errors as shown in Figures 6.1 and 6.2. The Personal COBOL Animator highlights each error line in the original code and coordinates the error line with the error messages listed in the Syntax Errors window shown in Figure 6.2. The error messages include the assigned number of the error and the error level. For example, 233-S means that this is error message 233 and its error level is Severe. The statement is ignored but the program cannot be compiled. The error message also contains a brief explanation of the error. Some of the errors will be immediately obvious; others may require you to look up the message number in Syntax Check Error Messages in Animator Help. This help feature is included in the Standard COBOL Reference entry in the Help menu. Still other errors may require you to seek help. As you progress through this book and gain practical experience, you will become increasingly self-sufficient.

Figure 6.2 Compilation Errors

Let us examine the errors:

233-S Unknown data description qualifier UNION

This error results from the first omitted hyphen in the definition of UNI UNION FEE in line 42; that is, the compiler does not know how to handle what it thinks are two data names in a row (UNI and UNION) and hence the error. In this case, the compiler has found a S-level error and ignores the rest of the statement. The compiler does not detect the missing hyphen between UNION and FEE. If only the first hyphen is inserted, on the next recompilation, the compiler will then discover the second hyphen is missing. Sometimes, one syntax error hides others so that the compiler is not able to detect them.

Correction: Insert hyphens to read UNI-UNION-FEE.

1014-E Period missing. Period assumed

A level number must follow a completed statement, but the period ending line 61 has been removed. In this instance, the compiler assumes that the period is present, so no harm is done, but it is poor programming to permit such E-level diagnostics to remain. Moreover, there are situations in which a missing period can be very damaging.

Correction: Insert a period at the end of line 61.

13-S User-name required

This error in line 124 is a subtle one that typically sends the beginner for help. START is intended as a paragraph name, and paragraph names must begin in the A-margin, so what's the problem? The difficulty is that START is a reserved word and cannot be used as a paragraph name.

Correction: Choose another name—for example, START-THE-PROGRAM.

12-S Operand STUDNET-FILE is not declared

The compiler was expecting a valid file name but didn't find one because line 144 references STUDNET-FILE rather than STUDENT-FILE. You know they are the same, but the compiler does not and hence the error.

Correction: Change the file name to STUDENT-FILE in statement 144.

564-S A scope-delimiter did not have a matching verb and was discarded.

This error in line 146 will disappear with the correction to the previous READ statement.

Correction: None required beyond the correction to line 144.

12-S Operand IND-ACTIVITY is not declared

The error is subtle because the program file contains IND-ACTIVITY-FEE in line 35, yet the data name IND-ACTIVITY appears on the listing and is flagged as an error. The problem is that the COMPUTE statement in line 159 extends beyond column 72, into columns 73–76, which are not interpreted by the compiler; that is, the compiler reads IND-ACTIVITY rather than IND-ACTIVITY-FEE.

Correction: Reformat the COMPUTE statement so that IND-ACTIVITY-FEE appears on the next line.

5-S User-name STU-CREDITS not unique

This message appears four times in a row and is associated with lines 163, 174, 176, and 179. This error message implies that two or more data names are the same; in this instance STU-CREDITS is defined in line 21 and again in line 81 (the latter should be DET-CREDITS), and the compiler does not know which is which.

Correction: Restore uniqueness to the data name in line 81, by changing STU-CREDITS to DET-CREDITS.

12-S Operand UNI-UNION-FEE is not declared

The error message references UNI-UNION-FEE as an undefined symbol and is another example of how one error can cause several others. Hyphens were omitted in the definition of UNI-UNION-FEE in line 42, and thus (as far as the compiler is concerned) the data name UNI-UNION-FEE does not exist.

Correction: This diagnostic will disappear with the correction to line 42.

34-S Operand UNI-SCHOLARSHIP should be numeric

Arithmetic is permitted only on numeric data names. UNI-SCHOLARSHIP, however, was defined in line 44 as an alphanumeric rather than a numeric data name, and hence the error.

Correction: Change the PICTURE clause in line 44 from X(6) to 9(6).

5-S User-name STU-CREDITS not unique

This error is identical to the earlier non-unique message from lines 163, 174, 175, 177, and 179. This error disguises another error. DET-CREDITS has not been defined and should be flagged. In this case, the same correction fixes both problems.

Correction: This error will disappear after changing STU-CREDITS to DET-CREDITS in line 81.

330-S Not a record name

A WRITE statement, such as the one in line 209, requires a record name rather than a file name.

Correction: Change line 209 to WRITE PRINT-LINE instead of WRITE PRINT-FILE.

12-S Operand UNI-UNION-FEE is not declared

This error is identical to the one in line 194 and is due to the omitted hyphens in the definition of UNI-UNION-FEE.

Correction: None required beyond the previous correction to line 42.

These are all of the compilation errors detected by the Animator. This example was prepared for an earlier edition of the book and a different compiler. In making the conversion to Personal COBOL we found two errors that the Animator did not flag. The first was in line 163, where there is no space between the “=” and PRICE-PER-CREDIT. COBOL requires spaces before and after arithmetic operators, but evidently the Animator tolerates this error. You should always make it a habit to put spaces before and after arithmetic operators. Other compilers will not be as forgiving.

The second error the Animator did not flag was in line 187. The MOVE statement moves the value of STU-SCHOLARSHIP (a four-position numeric field) to IND-SCHOLARSHIP (a three-position numeric field). The problem is that the sending field is larger than the receiving field, and thus the leftmost (most significant) digit may be truncated. This error could cause problems and should have had an I-level or possibly W-level message.

Common Compilation Errors

Compilation errors are a fact of life. Don't be discouraged if you have many compilation errors in your first few attempts, and don't be surprised if you have several pages of diagnostics. Remember that a single error in a COBOL program can result in many error messages, and that several errors often can be made to disappear with one correction. Before leaving the subject, it is worthwhile to review a list of common errors and suggested ways to avoid them:

Nonunique data names. This error occurs because the same data name is defined in two different records or twice within the same record. For example, CREDITS might be specified as an input field in STUDENT-FILE and again as output in a detail line. You can avoid the problem by prefixing every data name within a record by a unique prefix as shown below:

```
01 STUDENT-RECORD
   05 STU-NAME
       10 STU-LAST-NAME
       10 STU-INITIALS
   05 STU-CREDITS
   05 STU-UNION-MEMBER
   05 STU-SCHOLARSHIP
   05 STU-GPA
```

Omitted (or extra) periods. Every COBOL sentence should have a period. Omission in the first three divisions often results in the compiler's assumption of a period where one belongs, and such errors are generally harmless. The effect is far more serious in the Procedure Division, where missing and/or extra periods affect the generated logic.

Omitted space before or after an arithmetic operator. The arithmetic operators, **, *, /, +, and – all require a space before and after (a typical error for BASIC programmers, since the space is not required in that language).

Invalid picture clause for numeric entry. All data names used in arithmetic statements must have numeric picture clauses consisting of 9's, an implied decimal point, and an optional sign.

Conflicting picture and value clause. Numeric pictures must have numeric values (no quotes); nonnumeric pictures must have nonnumeric values (enclosed in quotes). Both entries below are *invalid*.

```
05 TOTAL    PIC 9(3)  VALUE '123'.
05 TITLE    PIC X(3)  VALUE 123.
```

Inadvertent use of COBOL reserved words. COBOL has a list of some 300 reserved words that can be used only in their designated sense; any other use results in one or several diagnostics. Some reserved words are obvious, for example, WORKING-STORAGE, IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE. Others—such as CODE, DATE, START, and REPORT—are less obvious. Instead of memorizing the list or continually referring to it, we suggest this simple rule of thumb: *Always use a hyphen in every data name you create.* This will work more than 99% of the time.

Conflicting RECORD CONTAINS clause and FD record description. This is a common error, even for established programmers. It can stem from careless addition in that the sum of the pictures in the FD does not equal the number of characters in the RECORD CONTAINS clause. It can also result from other errors within the Data Division, for example, when an entry containing a PICTURE clause is flagged. (Remember that if an E-level diagnostic occurs, that entry will be ignored, and the count is thrown off.)

Receiving field too small to accommodate sending field. This is an extremely common error, often associated with edited pictures (editing is discussed in Chapter 7). Consider the entries:

```
05 PRINT-TOTAL-PAY    PIC $$,$$$$.
05 WS-TOTAL-PAY       PIC 9(5).
```

```
MOVE WS-TOTAL-PAY TO PRINT-TOTAL-PAY.
```

The MOVE statement would generate the warning that the receiving field may be too small to accommodate the sending field. The greatest possible value for WS-TOTAL-PAY is 99,999; the largest possible value that could be printed by PRINT-TOTAL-PAY is \$9,999. Even though the picture for the print field contains five \$'s, one \$ must always be printed along with the numeric characters, hence the warning.

Omitted (or extra) hyphens in a data name. This is a careless error, but one that occurs too often. If, for example, we define PRINT-TOTAL-PAY in the Data Division and then reference PRINT TOTAL-PAY in the Procedure Division, the compiler catches the inconsistency. It doesn't state that a hyphen was omitted, but indicates that PRINT and TOTAL-PAY are undefined.

A related error is the insertion of extra hyphens where they don't belong, for example, WORKING-STORAGE-SECTION or DATA-DIVISION.

Misspelled data names or reserved words. Too many COBOL students are poor spellers. Sound strange? How do you spell *environment*? One or many errors can result, depending on which word was spelled incorrectly.

Reading a record name or writing a file name. The COBOL rule is very simple—read a file and write a record—but many people get it confused. Consider:

```
FD STUDENT-FILE
   DATA RECORD IS STUDENT-RECORD.
```

```
FD PRINT-FILE
   DATA RECORD IS PRINT-RECORD.
```

Correct entries:

```
READ STUDENT-FILE . . .
WRITE PRINT-RECORD . . .
```

Incorrect entries:

```
READ STUDENT-RECORD . . .
WRITE PRINT-FILE . . .
```

Going past column 72. This error can cause any of the preceding errors as well as a host of others. A COBOL statement must end in column 72 or before; columns 73–80 are left blank or used for program identification. (The 72-column restriction does not apply to data.)

Errors in Execution

After a program has been successfully compiled, it can proceed to execution, and therein lies the strength and weakness of the computer. The primary attractiveness of the machine is its ability to perform its task quickly; its weakness stems from the fact that it does exactly what it has been instructed to do. The machine cannot think for itself; the programmer must think for the machine. If you were to inadvertently instruct the computer to compute tuition by charging \$20 instead of \$200 per credit, then that is what it would do.

To give you an idea of what can happen, we have deliberately altered the original tuition billing program of Chapter 5 and created a new program, shown in Figure 6.3. Incorporated into this program are two types of errors: run time errors and logic errors. Run time errors prevent the program from carrying out its task even though the program compiled properly. Logic errors do not stop the program, but they cause invalid output from the program.

Figure 6.3 Tuition Billing Program with Execution Errors

```
1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. TUIT6EXE.
3 AUTHOR. CAROL VAZQUEZ VILLAR.
4
5 ENVIRONMENT DIVISION.
6 INPUT-OUTPUT SECTION.
7 FILE-CONTROL.
8 SELECT STUDENT-FILE ASSIGN TO 'A:\CHAPTR06\TUITIION.DAT'
9 ORGANIZATION IS LINE SEQUENTIAL.
10 SELECT PRINT-FILE
```

Figure 6.3 (continued)

```

11          ASSIGN TO PRINTER.
12
13 DATA DIVISION.
14 FILE SECTION.
15 FD STUDENT-FILE
16    RECORD CONTAINS 27 CHARACTERS.
17 01 STUDENT-RECORD.
18   05 STU-NAME.
19     10 STU-LAST-NAME    PIC X(15).
20     10 STU-INITIALS    PIC XX.
21   05 STU-CREDITS        PIC 9(2).
22   05 STU-UNION-MEMBER   PIC X.
23   05 STU-SCHOLARSHIP    PIC 9(4).
24   05 STU-GPA            PIC 999.
25
26 FD PRINT-FILE
27    RECORD CONTAINS 132 CHARACTERS.
28 01 PRINT-LINE          PIC X(132).
29
30 WORKING-STORAGE SECTION.
31 01 DATA-REMAINS-SWITCH PIC X(2) VALUE SPACES.
32
33 01 INDIVIDUAL-CALCULATIONS.
34   05 IND-TUITION        PIC 9(4) VALUE ZEROS.
35   05 IND-ACTIVITY-FEE   PIC 9(2) VALUE ZEROS.
36   05 IND-UNION-FEE      PIC 9(2) VALUE ZEROS.
37   05 IND-SCHOLARSHIP    PIC 9(4) VALUE ZEROS.
38   05 IND-BILL           PIC 9(6) VALUE ZEROS.
39
40 01 UNIVERSITY-TOTALS.
41   05 UNI-TUITION        PIC 9(6) VALUE ZEROS.
42   05 UNI-UNION-FEE      PIC 9(4) VALUE ZEROS.
43   05 UNI-ACTIVITY-FEE   PIC 9(4) VALUE ZEROS.
44   05 UNI-SCHOLARSHIP    PIC 9(6) VALUE ZEROS.
45   05 UNI-IND-BILL       PIC 9(6) VALUE ZEROS.
46
47 01 CONSTANTS-AND-RATES.
48   05 PRICE-PER-CREDIT   PIC 9(3) VALUE 200.
49   05 UNION-FEE          PIC 9(2) VALUE 25.
50   05 ACTIVITY-FEES.
51     10 1ST-ACTIVITY-FEE PIC 99  VALUE 25.
52     10 1ST-CREDIT-LIMIT PIC 99  VALUE 6.
53     10 2ND-ACTIVITY-FEE PIC 99  VALUE 50.
54     10 2ND-CREDIT-LIMIT PIC 99  VALUE 12.
55     10 3RD-ACTIVITY-FEE PIC 99  VALUE 75.

```

Implied decimal place missing

Figure 6.3 (continued)

```

56          05 MINIMUM-SCHOLAR-GPA PIC 9V9  VALUE 2.5.
57
58      01  HEADING-LINE.
59          05 FILLER                PIC X    VALUE SPACES.
60          05 FILLER                PIC X(12) VALUE 'STUDENT NAME'.
61          05 FILLER                PIC X(10) VALUE SPACES.
62          05 FILLER                PIC X(7)  VALUE 'CREDITS'.
63          05 FILLER                PIC X(2)  VALUE SPACES.
64          05 FILLER                PIC X(7)  VALUE 'TUITION'.
65          05 FILLER                PIC X(2)  VALUE SPACES.
66          05 FILLER                PIC X(9)  VALUE 'UNION FEE'.
67          05 FILLER                PIC X(2)  VALUE SPACES.
68          05 FILLER                PIC X(7)  VALUE 'ACT FEE'.
69          05 FILLER                PIC X(2)  VALUE SPACES.
70          05 FILLER                PIC X(11) VALUE 'SCHOLARSHIP'.
71          05 FILLER                PIC X(2)  VALUE SPACES.
72          05 FILLER                PIC X(10) VALUE 'TOTAL BILL'.
73          05 FILLER                PIC X(48) VALUE SPACES.
74
75      01  DETAIL-LINE.
76          05 FILLER                PIC X    VALUE SPACES.
77          05 DET-LAST-NAME          PIC X(15).
78          05 FILLER                PIC X(2)  VALUE SPACES.
79          05 DET-INITIALS          PIC X(2).
80          05 FILLER                PIC X(5)  VALUE SPACES.
81          05 DET-CREDITS           PIC 9(2).
82          05 FILLER                PIC X(6)  VALUE SPACES.
83          05 DET-TUITION           PIC 9(6).
84          05 FILLER                PIC X(7)  VALUE SPACES.
85          05 DET-UNION-FEE         PIC 9(3).
86          05 FILLER                PIC X(6)  VALUE SPACES.
87          05 DET-ACTIVITY-FEE     PIC 9(3).
88          05 FILLER                PIC X(8)  VALUE SPACES.
89          05 DET-SCHOLARSHIP      PIC 9(5).
90          05 FILLER                PIC X(6)  VALUE SPACES.
91          05 DET-IND-BILL         PIC 9(6).
92          05 FILLER                PIC X(49) VALUE SPACES.
93
94      01  DASH-LINE.
95          05 FILLER                PIC X(31) VALUE SPACES.
96          05 FILLER                PIC X(8)  VALUE ALL '-'.
97          05 FILLER                PIC X(2)  VALUE SPACES.
98          05 FILLER                PIC X(8)  VALUE ALL '-'.
99          05 FILLER                PIC X(2)  VALUE SPACES.
100         05 FILLER                PIC X(7)  VALUE ALL '-'.

```

Figure 6.3 (continued)

```

101      05 FILLER                PIC X(6) VALUE SPACES.
102      05 FILLER                PIC X(7) VALUE ALL '-'.
103      05 FILLER                PIC X(5) VALUE SPACES.
104      05 FILLER                PIC X(7) VALUE ALL '-'.
105      05 FILLER                PIC X(49) VALUE SPACES.
106
107      01 TOTAL-LINE.
108          05 FILLER                PIC X(8) VALUE SPACES.
109          05 FILLER                PIC X(17)
110              VALUE 'UNIVERSITY TOTALS'.
111          05 FILLER                PIC X(8) VALUE SPACES.
112          05 TOT-TUITION          PIC 9(6).
113          05 FILLER                PIC X(6) VALUE SPACES.
114          05 TOT-UNION-FEE       PIC 9(4).
115          05 FILLER                PIC X(5) VALUE SPACES.
116          05 TOT-ACTIVITY-FEE    PIC 9(4).
117          05 FILLER                PIC X(7) VALUE SPACES.
118          05 TOT-SCHOLARSHIP     PIC 9(6).
119          05 FILLER                PIC X(6) VALUE SPACES.
120          05 TOT-IND-BILL        PIC 9(6).
121          05 FILLER                PIC X(49) VALUE SPACES.
122
123      PROCEDURE DIVISION.
124      PREPARE-TUITION-REPORT.
125          OPEN INPUT STUDENT-FILE
126              OUTPUT PRINT-FILE.
127          PERFORM WRITE-HEADING-LINE.
128          PERFORM READ-STUDENT-FILE.
129          PERFORM PROCESS-STUDENT-RECORD
130              UNTIL DATA-REMAINS-SWITCH = 'NO'.
131          PERFORM WRITE-UNIVERSITY-TOTALS.
132          CLOSE STUDENT-FILE
133              PRINT-FILE.
134          STOP RUN.
135
136      WRITE-HEADING-LINE.
137          MOVE HEADING-LINE TO PRINT-LINE.
138          WRITE PRINT-LINE
139              AFTER ADVANCING PAGE.
140          MOVE SPACES TO PRINT-LINE.
141          WRITE PRINT-LINE.
142
143      READ-STUDENT-FILE.
144          READ STUDENT-FILE
145              AT END MOVE 'NO' TO DATA-REMAINS-SWITCH

```

Figure 6.3 (continued)

```
146         END-READ.
147
148     PROCESS-STUDENT-RECORD.
149         PERFORM READ-STUDENT-FILE.
150         PERFORM COMPUTE-INDIVIDUAL-BILL.
151         PERFORM INCREMENT-UNIVERSITY-TOTALS
152         PERFORM WRITE-DETAIL-LINE.
153
154     COMPUTE-INDIVIDUAL-BILL.
155         PERFORM COMPUTE-TUITION.
156         PERFORM COMPUTE-UNION-FEE.
157         PERFORM COMPUTE-ACTIVITY-FEE.
158         PERFORM COMPUTE-SCHOLARSHIP.
159         COMPUTE IND-BILL = IND-TUITION + IND-UNION-FEE +
160             IND-ACTIVITY-FEE - IND-SCHOLARSHIP.
161
162     COMPUTE-TUITION.
163         COMPUTE IND-TUITION = PRICE-PER-CREDIT * STU-CREDITS.
164
165     COMPUTE-UNION-FEE.
166         IF STU-UNION-MEMBER = 'Y'
167             MOVE ZERO TO IND-UNION-FEE
168         ELSE
169             MOVE UNION-FEE TO IND-UNION-FEE
170         END-IF.
171
172     COMPUTE-ACTIVITY-FEE.
173         EVALUATE TRUE
174             WHEN STU-CREDITS <= 1ST-CREDIT-LIMIT
175                 MOVE 1ST-ACTIVITY-FEE TO IND-ACTIVITY-FEE
176             WHEN STU-CREDITS > 1ST-CREDIT-LIMIT
177                 AND STU-CREDITS <= 2ND-CREDIT-LIMIT
178                 MOVE 2ND-ACTIVITY-FEE TO IND-ACTIVITY-FEE
179             WHEN STU-CREDITS > 2ND-CREDIT-LIMIT
180                 MOVE 3RD-ACTIVITY-FEE TO IND-ACTIVITY-FEE
181             WHEN OTHER
182                 DISPLAY 'INVALID CREDITS FOR: ' STU-NAME
183         END-EVALUATE.
184
185     COMPUTE-SCHOLARSHIP.
186         IF STU-GPA > MINIMUM-SCHOLAR-GPA
187             MOVE STU-SCHOLARSHIP TO IND-SCHOLARSHIP
188         ELSE
189             MOVE ZERO TO IND-SCHOLARSHIP
```

READ statement incorrectly placed

Order of IF statement is reversed

Figure 6.3 (continued)

```

190         END-IF.
191
192     INCREMENT-UNIVERSITY-TOTALS.
193         ADD IND-TUITION      TO UNI-TUITION.
194         ADD IND-ACTIVITY-FEE TO UNI-ACTIVITY-FEE.
195         ADD IND-SCHOLARSHIP TO UNI-SCHOLARSHIP.
196         ADD IND-BILL        TO UNI-IND-BILL.
197
198     WRITE-DETAIL-LINE.
199         MOVE STU-LAST-NAME TO DET-LAST-NAME.
200         MOVE STU-INITIALS TO DET-INITIALS.
201         MOVE STU-CREDITS TO DET-CREDITS.
202         MOVE IND-TUITION TO DET-TUITION.
203         MOVE IND-UNION-FEE TO DET-UNION-FEE.
204         MOVE IND-ACTIVITY-FEE TO DET-ACTIVITY-FEE.
205         MOVE IND-SCHOLARSHIP TO DET-SCHOLARSHIP.
206         MOVE IND-BILL TO DET-IND-BILL.
207         MOVE DETAIL-LINE TO PRINT-LINE.
208         WRITE PRINT-LINE
209             AFTER ADVANCING 1 LINE.
210
211     WRITE-UNIVERSITY-TOTALS.
212         MOVE DASH-LINE TO PRINT-LINE.
213         WRITE PRINT-LINE.
214         MOVE UNI-TUITION TO TOT-TUITION.
215         MOVE UNI-UNION-FEE TO TOT-UNION-FEE.
216         MOVE UNI-ACTIVITY-FEE TO TOT-ACTIVITY-FEE.
217         MOVE UNI-SCHOLARSHIP TO TOT-SCHOLARSHIP.
218         MOVE IND-BILL TO TOT-IND-BILL.
219         MOVE TOTAL-LINE TO PRINT-LINE.
220         WRITE PRINT-LINE
221             AFTER ADVANCING 1 LINE.

```

ADD statement is missing for UNI-UNION-FEE

Wrong field is moved to print line

File Status Codes

Input/Output operations occur throughout the execution of a COBOL program and consequently are a source of frequent run-time errors. The window shown in Figure 6.4a represents one of the most common types of errors—attempting to read from a file that doesn't exist.

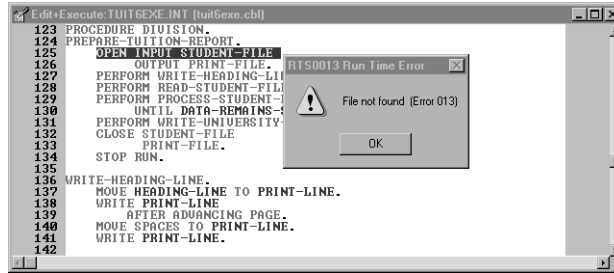
After we click on OK, the Animator shows us the line where the error was detected, as in Figure 6.4b. The OPEN statement relates to the SELECT ... ASSIGN statements in the Environment Division. The COBOL SELECT statement ties a programmer-chosen file name to an implementor name. In Windows-based COBOL compilers the ASSIGN clause allows the definition of a file name. The combination of SELECT and ASSIGN associates a COBOL file, such as STUDENT-FILE, with a file on disk such as TUITION.DOT in Figure 6.4c, line 8.

The problem is that TUITION.DOT does not exist; look carefully at the properties of TUITION in Figure 6.4d. The file extension is *DAT* rather than *DOT*. In other words, the COBOL program is attempting to read from a file that isn't there, an impossible situation for the program that leads to an execution or *run time system* (RTS) error.

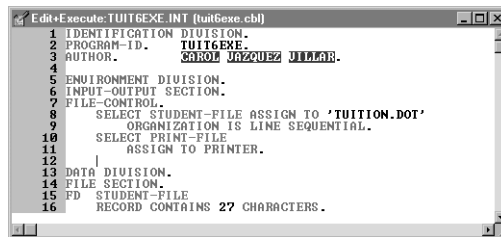
Figure 6.4 File Status Errors



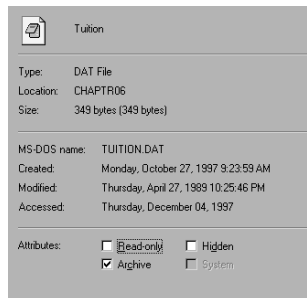
(a) File not found message



(b) Location where error detected.



(c) Error in SELECT statement



(d) Tuition File Properties

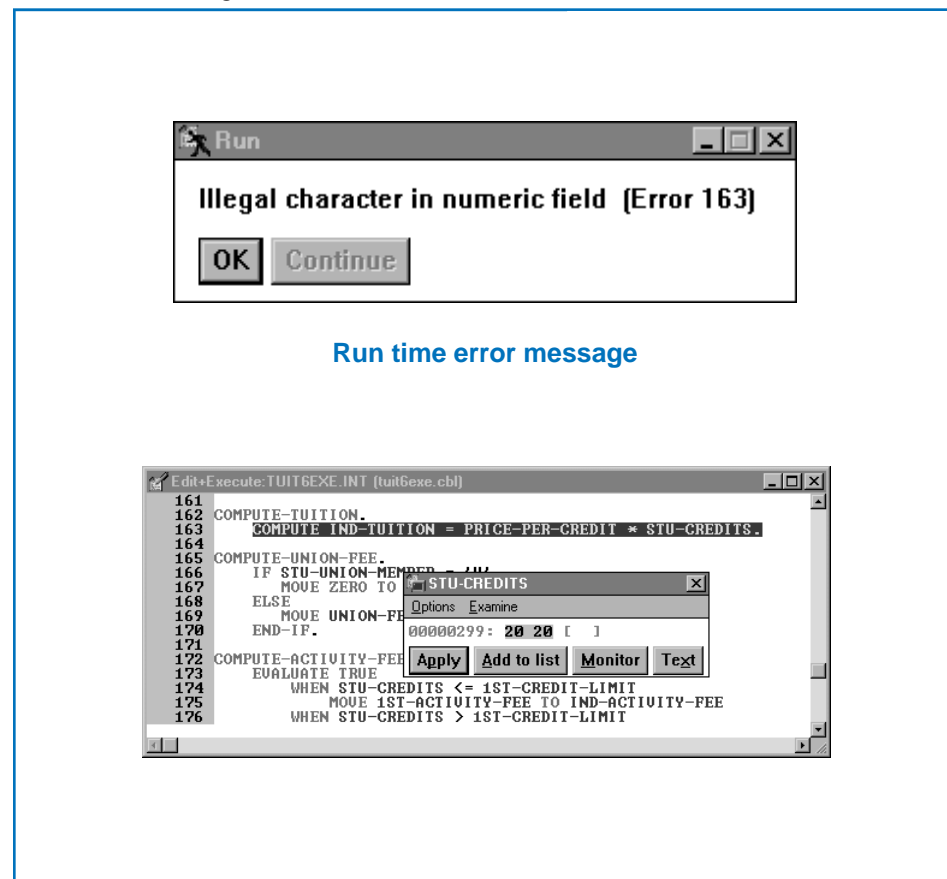
The Animator is very picky about file names. For example, even if you spell the name correctly but do not correctly specify the path, the Animator may not be able to find the file. If the program specifies only the name and extension, but the actual file is in a working directory other than that expected by the Animator, the program will not be able to find the file. Appendix B shows you how to make sure that the Animator knows where to look for files.

Another Run Time Error

After correcting the file name, we recompiled the program and ran it again, this time with the results in Figure 6.5. Figure 6.5a shows another common RTS error, “Illegal Character in Numeric Field.” This error almost always comes from having spaces in a numeric field. One cause of those spaces is when the program reads a data file that actually has spaces in the field. In other words, if the field has a PICTURE of 9(5) and the actual contents are “123”, the program will fail if it tries to use this field in a computation. This problem is a data problem rather than a program problem. When you create test data, be sure to type in leading zeros for any numeric fields. In this case the field should have been “00123”.

A second reason for spaces in a numeric field is when the program attempts to read beyond the end of a file. The incorrect placement of the READ in line 149 of Figure 6.3 causes this condition. Figure 6.5b shows the line where the error was detected. We clicked on each data name in the statement, and the Animator showed us the current contents of the field. STU-CREDITS contains only spaces and caused the error. In this case, when the program read beyond the end of the file, COBOL inserted spaces into STUDENT-RECORD (line 17), including STU-CREDITS. To correct the problem we restored the READ to the end of PROCESS-STUDENT-RECORD, recompiled the program, and reran it. This action corrects the run-time errors, but there are still logic errors to deal with.

Figure 6.5 Illegal Character Run Time Error



Logic Errors

There are several subtle errors in Figure 6.6:

1. The university total for union fees is zero rather than a computed amount.
2. The sum of the individual fills in the total line appears as 850 (the amount for the last record), rather than a running total of 24550.
3. The union fees are reversed for each student. For example, James and Baker are charged \$25 when they should be charged nothing; conversely, Part-Timer and Jones are charged nothing when their fee is \$25.
4. James was erroneously awarded a scholarship of \$500; James, however, does not qualify because his average is below 2.5.

We emphasize that these logic errors are not contrived but are typical of students and beginning programmers. Even the accomplished practitioner can be guilty of similar errors when rushed or careless. Realize also that logic errors occur without fanfare. There are no compiler diagnostics or RTS error messages to warn of impending trouble. The program has compiled cleanly and runs smoothly to the end; there is nothing to indicate a problem.

The errors in Figure 6.3 are errors in execution, rather than in compilation. The program compiled cleanly because it is *syntactically correct*, but it executed improperly because it is *logically incorrect*. Nevertheless, the program did precisely what it was instructed to do, which, unfortunately, is not what the programmer wanted it to do. It is necessary, therefore, to find the source of each logic error, as discussed below.

1. The totals for the university are computed in the paragraph INCREMENT-UNIVERSITY-TOTALS (lines 192–196), in which the individual amounts for the student being processed are added to the running university totals. Note, however, that the ADD statement for UNI-UNION-FEE is conspicuously absent, and hence the value of UNI-UNION-FEE remains unchanged throughout the program.
2. UNI-IND-BILL is defined in line 45 and correctly incremented for each record in line 196; so far, so good. However, when the total line is built in line 218, IND-BILL rather than UNI-IND-BILL is moved to TOT-IND-BILL, causing the individual last bill (for Kerbel) to be printed as the total.
3. IND-UNION-FEE is calculated in a simple IF statement in lines 166–170, in which the IF and ELSE clauses are reversed; that is, the union fee is \$25 for students who belong to the union as indicated by a Y in the appropriate incoming field.
4. The definition STU-GPA in line 24 incorrectly omits the implied decimal point in the PICTURE clause. Hence all incoming averages will be interpreted as ten times their true value (i.e., 2.5 will be stored as 25). Thus, all students will have an average greater than 2.5, and hence all students with potential scholarships will receive the award.

Figure 6.6 Tuition Billing Report Comparisons—Invalid and Valid

STUDENT NAME	CREDITS	TUITION	UNION FEE	ACT FEE	SCHOLARSHIP	TOTAL BILL	
SMITH	JB	15	003000	000	075	00000	003075
JAMES	HR	15	003000	025	075	00500	002600
BAKER	SR	09	001800	025	050	00500	001375
PART-TIMER	JR	03	000600	000	025	00000	000625
JONES	PL	15	003000	000	075	00000	003075
HEAVYWORKER	HM	18	003600	025	075	00000	003700
LEE	BL	18	003600	025	075	00000	003700
CLARK	JC	06	001200	025	025	00000	001250
GROSSMAN	SE	07	001400	025	050	00000	001475
FRANKEL	LF	10	002000	025	050	00000	002075
BENWAY	CT	03	000600	025	025	00250	000400
KERBEL	NB	04	000800	025	025	00000	000850
UNIVERSITY TOTALS		024600	0000	0625	001250	000850	

(a) Invalid Output

Annotations for (a):

- 3 Union fees are reversed (points to 000 in Union Fee for SMITH)
- 4 Should not have a scholarship (points to 00000 in Scholarship for SMITH)
- 2 Total is incorrect (points to 000850 in Total Bill for UNIVERSITY TOTALS)
- 1 Union fee was not summed (points to 0000 in Union Fee for UNIVERSITY TOTALS)

STUDENT NAME	CREDITS	TUITION	UNION FEE	ACT FEE	SCHOLARSHIP	TOTAL BILL	
SMITH	JB	15	003000	025	075	00000	003100
JAMES	HR	15	003000	000	075	00000	003075
BAKER	SR	09	001800	000	050	00500	001350
PART-TIMER	JR	03	000600	025	025	00000	000650
JONES	PL	15	003000	025	075	00000	003100
HEAVYWORKER	HM	18	003600	000	075	00000	003675
LEE	BL	18	003600	000	075	00000	003675
CLARK	JC	06	001200	000	025	00000	001225
GROSSMAN	SE	07	001400	000	050	00000	001450
FRANKEL	LF	10	002000	000	050	00000	002050
BENWAY	CT	03	000600	000	025	00250	000375
KERBEL	NB	04	000800	000	025	00000	000825
UNIVERSITY TOTALS		024600	0075	0625	000750	024550	

(b) Valid Output (from Chapter 5)

Tips for Debugging

It was easy to find the execution errors just discussed because we created them in the first place, and hence we knew exactly where to look. In practice, however, it is not so easy. Fortunately, the Personal COBOL Animator provides some powerful tools to help

with debugging programs. Appendix A gives an extended discussion of the Animator and even has a short debugging tutorial. However, we can provide a few tips here as well.



1. Step through the program using the Watch Button. This button shows the contents of each field in the current statement.
2. Use breakpoints to stop the program at critical junctures in the program. By using breakpoints, you can run the program at full speed until the breakpoint is encountered and then step through the questionable code.
3. To save paper, write your output to the screen rather than the printer. When you are actually ready to print the report, a simple change to the ASSIGN will accomplish the task. For example:
To print to screen: SELECT PRINT-FILE ASSIGN TO PRINTER 'CON'.
To print to printer: SELECT PRINT-FILE ASSIGN TO PRINTER.
4. Double-clicking on any data name will bring up the current value of the field, whether it is in the current execution line or not.
5. Using the Find option in the Edit Menu will highlight all occurrences of a data name in the program.

There are many other features in the Animator to help with debugging, and as you gain proficiency in programming you will see how to use them.

DISPLAY Statement

It is often helpful to display intermediate results of a program as the program is being executed. One way to accomplish this is through the insertion of **DISPLAY** statements at strategic points in the program. The statement enables you to print the value of one or more data names and/or one or more literals without having to format a record description. Consider:

$$\text{DISPLAY } \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-2} \end{array} \right\} \right] \dots$$

The DISPLAY statement produces the contents of each item listed in the order shown. For example,

1. DISPLAY STUDENT-RECORD.
2. DISPLAY 'Record being processed: ' STUDENT-RECORD.
3. DISPLAY 'COMPUTE-TUITION paragraph is entered'
4. DISPLAY 'Student data: ' STU-NAME STU-CREDITS.

Examples one and two both display the value of the data name STUDENT-RECORD; the second example, however, precedes the data name with a literal to facilitate interpretation of the output. Example three displays just a literal but could be used (in conjunction with similar DISPLAY statements in other paragraphs) to show the flow of program execution. Example four displays a literal and two data names.

The Structured Walkthrough

Although it is reasonable to expect errors, the programmer is also expected (reasonably) to find and correct them. Until recently, error detection and correction was a lonely activity. A programmer was encouraged to *desk check*—that is, read and reread the code—in an attempt to discern logical errors before they occurred. Desk checking is still an important activity, but it is frequently supplemented by a newer technique, the *structured walkthrough*.

The walkthrough brings the evaluation into the open. It requires a programmer to have his or her work reviewed formally and periodically by a peer group. The theory is simple—a programmer is too close to his or her work to see potential problems adequately and evaluate them objectively. The purpose of the walkthrough is to ensure that all specifications are met, and that the logic and its COBOL implementation are correct.

The earlier an error is found, the easier it is to correct and thus the single most important objective of a walkthrough is *early error detection*. Walkthroughs occur at several stages during a project, beginning in the analysis phase, where the purpose is to ensure that the systems analyst has understood the user's requirements. Walkthroughs occur again during the design phase, after the programmer has developed a hierarchy chart and/or associated pseudocode. Finally, walkthroughs occur during the implementation phase, during which the programmer presents actual code prior to testing.

Walkthroughs are scheduled by the person being reviewed, who also selects the reviewers. The programmer distributes copies of the work (for example, a hierarchy chart, pseudocode, or a COBOL program) prior to the session. Reviewers are supposed to study the material in advance so that they can discuss it intelligently. At the walkthrough itself, the programmer presents the material objectively, concisely, and dispassionately. He or she should encourage discussion and be genuinely glad when errors are discovered.

One of the reviewers should function as a *moderator* to keep the discussion on track. Another should act as a *secretary* and maintain an *action list* of problems uncovered during the session. At the end of the walkthrough the action list is given to the programmer, who in turn is expected to correct the errors and notify attendees accordingly. The objective of the walkthrough is to find errors, not to correct them. The latter is accomplished by the programmer upon receipt of the action list.

The preceding discussion may read well in theory, but programmers often dislike the walkthrough concept. The probable reason is that they dislike having their work reviewed and regard criticism of code as a personal affront, intended or otherwise. This attitude is natural and stems from years of working as individuals.

In addition, walkthroughs can and have become unpleasant and ego-deflating experiences. “Structured walkover” and “stomp through” are terms that have been applied to less-than-successful sessions. Only if the atmosphere is kept open and nondefensive, only if the discussion is restricted to major problems rather than trivial errors, and only if personality clashes are avoided can the walkthrough be an effective technique. To have any chance of success, programmers who function as both reviewer and reviewee must adhere to the following guidelines:

1. *The program, and not the programmer, is reviewed.* Structured walkthroughs are intended to find programming problems; they will not be used by management as an evaluation tool. No one should keep count of how many

errors are found in an individual's work or how many errors one finds in someone else's. It is quite logical, therefore, to exclude the project manager—that is, the individual in charge of salaries and promotions—from review sessions.

2. *Emphasis is on error detection, not correction.* It is assumed that the individual being reviewed will take the necessary corrective action. Reviewers should not harp on errors by discussing how to correct them; indeed, no corrections whatever are made during a walkthrough.
3. *Everyone, from senior analyst to trainee, has his or her work reviewed.* This avoids singling out an individual and further removes any stigma from having one's work reviewed. It also promotes the give-and-take atmosphere that is so vital to making the concept work.
4. *A list of well-defined objectives for each session should be specified in advance.* Adherence to this guideline keeps the discussion on track and helps to guarantee productive discussions. Another guideline is to impose a predetermined time limit, from half an hour to two hours. Walkthroughs will eventually cease to be productive and degenerate into a discussion of last night's ball game, the new manager, the latest rumor, or some other "hot" topic. The situation should be anticipated and avoided, perhaps by scheduling walkthroughs an hour before lunch. If all of the walkthrough's objectives have not been met when the deadline is reached, schedule a second session.
5. *Participation must be encouraged and demanded from the reviewers.* A walkthrough will indeed become a waste of time if no one has anything to say. Let it be known in advance that each reviewer will be expected to make at least two comments, one positive and one negative. Alternatively, require each reviewer to come to the session with a list of at least three questions.

S U M M A R Y

Points to Remember

- Compilation errors occur in the translation of COBOL to machine language and result from a violation of COBOL syntax—for example, a misspelled data name or an entry in the wrong column.
- Run time and execution errors develop after compilation has taken place, and are caused by improper logic and/or improper COBOL implementation of valid logic.
- A program may compile cleanly and be logically correct, yet still fail to execute if there are problems with the associated data files. Run time errors will occur and generate RTS error messages to help determine the cause of such data management errors.
- Sometimes data file problems are not the fault of the program, but are from the data file itself. The most common problem occurs when a numeric field includes spaces rather than zeroes.

- The Animator provides many tools for debugging and can be quite helpful in tracking both syntax and logic errors.
- A structured walkthrough is an open evaluation of an individual's work by a group of his or her peers, with the primary objective of detecting errors as soon as possible in the development cycle.

Key Words and Concepts

Action list	Execution error
Compilation error	File status codes
Compiler option	Interactive debugger
Cross-reference listing	Moderator
Debugging	Run Time System
Desk checking	Secretary
Early error detection	Structured walkthrough

COBOL Element

DISPLAY

F I L L - I N

- _____ errors occur in the translation of COBOL to machine language.
- _____ errors occur after a program has been successfully translated to machine language.
- Incorrect translation of valid pseudocode into COBOL will most likely produce _____ errors.
- Misspelling a reserved word will most likely produce a _____ error.
- If a program _____ cleanly, it means only that the program has been successfully translated into machine language.
- _____ errors are accompanied by some type of error message, whereas _____ errors are frequently undetected by the computer.
- The process of peer review is known as a _____.
- The errors that are detected during a _____ are entered on an _____, which is maintained by the secretary.
- The emphasis in a structured walkthrough is on error _____, not error _____.
- One suggestion for conducting successful walkthroughs is to remember that the _____, and not the _____, is reviewed.

11. _____ cause a program to stop processing even though it is syntactically correct.
12. _____ are helpful in detecting errors in execution that pertain to data management.

TRUE / FALSE

1. If a program compiles with no diagnostics, it must execute correctly.
2. If a program compiles with warning diagnostics, execution will be suppressed.
3. If a program contains logical errors but not syntactical errors, the compiler will print appropriate warnings.
4. A COBOL program is considered data by the COBOL compiler.
5. An error in one COBOL statement can cause errors in several other, apparently unrelated, statements.
6. There are several different levels (of severity) of compilation errors.
7. Paragraph names begin in the A margin.
8. Spaces are required before and after arithmetic symbols.
9. Spaces are required before and after punctuation symbols.
10. A data name that appears in a COMPUTE statement can be defined with a picture of X's.
11. Data names may contain blanks.
12. The contents of columns 73–80 are ignored by the compiler.
13. In a COBOL program one reads a record name and writes a file name.
14. The emphasis in a structured walkthrough is on error detection rather than error correction.
15. Walkthroughs should be held for trainees only, as these are the individuals most likely to make mistakes.
16. Managers typically do not attend walkthroughs.
17. A walkthrough generally takes a minimum of two hours.
18. Walkthroughs should be restricted to the coding phase of a project.

PROBLEMS

1. Has your work ever been the subject of a structured walkthrough? Was the experience helpful or a waste of time, or worse? Are you looking forward to your next walkthrough?
2. Do you agree with banning managers from walkthroughs? Is it possible that the role of moderator in a walkthrough might best be filled by the project manager?

3. Do you agree with the authors' suggestions for successful walkthroughs? Are there any guidelines you wish to add to the list? To remove from the list?
4. Identify the syntactical errors in the COBOL fragment in Figure 6.8.
5. Identify the logical errors in the COBOL fragment in Figure 6.9. (Assume there are no other READ statements in the program.)
6. The COBOL fragment in Figure 6.10a is taken from a program that compiled cleanly but failed to execute. The error message is in Figure 6.10b. Explain the problem.

Figure 6.8 COBOL Fragment for Problem 4

```

IDENTIFICATION DIVISION.
PROGRAM ID.  ERRORS.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
SELECT EMPLOYEE-FILE  ASSIGN TO 'A:\CHAPTR06\EMP.DAT'
      ORGANIZATION IS LINE SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  EMPLOYEE-FILE
   RECORD CONTAINS 50 CHARACTERS
   DATA RECORD IS EMPLOYEE-RECORD.
   EMPLOYEE-RECORD.
   05  EMP-NAME                PIC X(20).
   05  EMP-NUMBER              PIC X(9).
   05  FILLER                  PIC X(20).
WORKING STORAGE SECTION.
10  END-OF-FILE-SWITCH        PIC X(3)  VALUE BLANKS.

```

Figure 6.9 COBOL Fragment for Problem 5

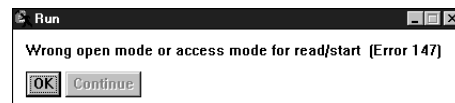
```

WORKING-STORAGE SECTION.
01  END-OF-FILE-SWITCH  PIC X(3)  VALUE 'YES'.
.
.
PROCEDURE DIVISION.
MAINLINE.
.
.
   PERFORM PROCESS-RECORDS
      UNTIL END-OF-FILE-SWITCH = 'YES'
.
.
PROCESS-RECORDS.
   READ EMPLOYEE-FILE
      AT END MOVE 'YES' TO END-OF-FILE-SWITCH
   END-READ.

```

Figure 6.10 COBOL Fragment for Problem 6

```
SELECT STUDENT-FILE  ASSIGN TO 'A:\CHAPTR06\TUITION.DAT'  
      ORGANIZATION IS LINE SEQUENTIAL.  
SELECT PRINT-FILE  
      ASSIGN TO PRINTER.  
.  
.  
PROCEDURE DIVISION.  
PREPARE-SENIOR-REPORT.  
  READ STUDENT-FILE  
    AT END MOVE 'NO' TO DATA-REMAINS-SWITCH  
  END-READ.  
  OPEN INPUT  STUDENT-FILE  
    OUTPUT PRINT-FILE.  
  PERFORM WRITE-HEADING-LINE.  
  PERFORM PROCESS-RECORDS  
    UNTIL DATA-REMAINS-SWITCH = 'NO'.  
  CLOSE STUDENT-FILE  
    PRINT-FILE.  
STOP RUN.
```

(a) COBOL Fragments**(b) Error Message**

