The environment we discuss in this book is the software environment on the mainframe. We note here that the hardware environment has been changing continuously, and what we have today does not even closely resemble the hardware systems of the past. The original mainframe computers were behemoths occupying large amounts of floor space, and now, as we all know, some of the personal computers of today have just as much power.

Mainframes have been developed almost exclusively by the IBM Corporation. The operating system has evolved from the OS/360, OS/370, and MVS/370 to the current OS/390. Although the software has also changed, the basic flavor has been retained. For example, when I started my work on the mainframe, SPF was the file editor and it continues to be so today, even though a lot of new functionality has been added.
COMPUTERS, USERS, AND DEVELOPERS

When I entered the field, computers were either the large mainframes from IBM or Amdahl or the so-called minicomputers from companies like DEC, HP, and UNISYS. Personal computers had not yet been introduced. Big companies used mainframe computers to do their larger applications, while minicomputers were used for specific tasks such as data entry or for scientific applications.

In the 1970s, programmers typically wrote code on coding sheets, which had the columns marked on them. They gave these coding sheets to data entry clerks, who punched them onto punched cards. Today the first six columns of COBOL code are used for sequence numbers, and this dates back to the punched-card days. If you ever dropped your deck of punched cards, you could put it into a mechanical device that would physically sort it for you. That was often a lifesaver.

Eventually the punched cards gave way to terminals, which were used to access interactive facilities such as TSO (time share option) and Roscoe (a facility similar to TSO, but not as powerful). TSO was soon enhanced with SPF.

All the terminals attached to the mainframe IBM computers were kept in air-conditioned rooms with raised floors so that the wires could be kept away from people traffic. This soon changed, with everyone having a terminal at his or her desk. Initially, these were dumb terminals that could only be used to interact with the mainframe via the keyboard. They could not do anything locally except display information. The terminals were kept close to controlling devices, and groups of controlling devices talked to the computer. Today, dumb terminals have given way to PCs, so that most developers have a PC on their desk which can be used for connection to the mainframe as well as for word processing, e-mail, and LAN and internet connectivity.

In the early days, disk storage was at a premium. Memory space was limited and the load modules had to be kept small (4K) in order to optimize the execution. Many of the coding decisions made in programs written in the early days reflected these limitations, most of which have long since disappeared.

The mainframe environment can be looked at from two perspectives—that of the end users of the applications developed on the system and that of the developers who are developing applications in this environment. From either perspective, this environment is huge. Thousands and thousands of applications have been developed using the mainframe computers. As a matter of fact, with the impending Year 2000 problem, it is estimated that over 50 billion lines of code on the mainframe have to be scanned and corrected.

A phenomenally large number of users are accessing databases that hold millions and millions of records. For example, billing systems of telephone companies and credit card companies hold a large amount of information about their customers. Government agencies, too, have gigantic databases. In order to process all this data, we definitely need mainframe computers, which have become the workhorses of large corporations, and we need software, appropriately coded.
Over the past 30 years or so, hundreds of thousands of software developers have been busy writing code. Thousands of developers from all corners of a company access the mainframe computer. The mainframe processor(s) serves each one of them very well—so much so that, when you are working on the mainframe computer, you almost feel that you are the only one accessing the machine. Sometimes it is almost like having your own PC.

In production environments, mainframes serve a large number of users, and in the development environment they serve a large number of developers. This makes the mainframe environment very useful for large corporations. Also, mainframe computers came long before PCs, and we have invested billions and billions of dollars in developing programs for them. It is not easy to get rid of these applications overnight.

In the mainframe environment thousands of users are logged onto one computer, which usually is situated in some faraway place, or at least on a different floor. If a PC breaks down, we can easily reboot it. This is not the case in a mainframe environment, owing to the large number of users logged on. Unless there is a real disaster, a mainframe computer never gets rebooted during a working day.

When a PC breaks down, the single user is affected. When a minicomputer breaks down, a whole floor of people may be affected. But when a mainframe computer breaks down, users from a whole building, or many buildings, can be affected. I remember working on an Accounts Payable System in the mainframe environment where hundreds of clerks who entered data into the system would become idle whenever our application went down. This means, for one thing, that we cannot just restart a mainframe computer like a PC, and for another, that we have to keep careful track of the system’s availability. This is one main difference between the mainframe environment and other platforms.

Each user in the mainframe environment has a userid, and this is the only way to log onto the mainframe. Every userid can access only the resources belonging to it. It can access common files only if explicit permission has been given. This can be done because resource access can be centrally controlled. Security can be pretty tight on mainframe computers. Also, in general, it is not easy to communicate between mainframe computers.

**DEVELOPMENT ENVIRONMENT**

**File Manipulations**

In order to function in the mainframe environment, one has to be able to write programs in COBOL, compile and link these programs, create execution JCL, and submit jobs. This is the bulk of a developer’s work. In earlier days, programmers usually also did the analysis, and most of us were called programmer-analysts.

Creation of programs involves manipulation of files, so the first thing one has to learn is how to manipulate files. File manipulation is the main function of ISPF—
Interactive System Productivity Facility. This name has evolved over a period of time from its earlier forms, Structured Programming Facility and System Productivity Facility. ISPF is the main file-manipulation utility, still going strong and continuing to evolve with time. It was one of the best products available when it was first introduced, and it is still one of the best for its purposes. There were no Windows or Mac graphic interfaces in those days. Yet in ISPF we could work using multiple screens (today we would call them windows). In some sense, the split-screen mode of ISPF is the precursor to the windows-based systems.

We consider now the concept of files themselves. Files in the mainframe environment are defined by strict rules. A file consists of records. A record can be of either fixed or variable length and can be grouped into blocks. In the mainframe lingo files are referred to as data sets, and they can be sequential or partitioned data sets (PDS). One can create, edit, and delete data sets using ISPF menus. In order to create data sets you must use either ISPF or JCL. Also you must know the attributes of each record and block. In a sense you have to understand your files well, if you want to process them.

All files are character oriented, and the 80-column records that you so often see are based on the old punched cards. ISPF is oriented to a 20-×-80-character-based screen. The 80 made it compatible with punched cards in the early days.

Programming Languages

The preferred programming language was and still is COBOL (Common Business Oriented Language). Predictions of COBOL’s disappearing and giving way to sophisticated block-structured languages have not yet materialized. Why has COBOL been preferred over the scientific language FORTRAN (Formula Translation)? The reason is its English-like syntax. COBOL was developed specifically for business applications, where there were more analysts who had to understand the code. Proper use of variable names and paragraph names in COBOL can generate code that is very easy for business analysts to understand. COBOL programmers were trained easily by technical institutes, while universities stuck to their FORTRANs, ALGOLs, and SNOBOLs. COBOL was more useful for character-processing business applications than for scientific number crunching. With the advent of block-structured languages came IBM’s PL/1 language, which includes the features of COBOL and FORTRAN along with some features of the block-structured languages. COBOL II added many features of block-structured languages to COBOL, COBOL, FORTRAN, and PL/1 have remained the main languages of the mainframe environment.

JCL

One of the main ways to interact with the mainframe computers is through Job Control Language (JCL). We assumed that JCL would become a thing of the past as soon as
computers got large enough to do everything on line. This, of course, has not happened, and it is not likely to happen any time soon. JCL controls resources such as data sets and devices such as tape drives and disk storage. A program called RACF (Resource Access Control Facility) controls access to these data sets. You can create, delete, and copy data sets using JCL.

JCL is also used for batch jobs. These are jobs that run in the background, often overnight, and update files or databases. Batch jobs are the backbone of large application systems. In practice, if you think about it, it is usually not necessary to update the master file instantaneously. So we could collect all the updateable information during the day and update the master file at night. In practice, this is what happens in all big corporations. The master files get updated via batch jobs. You can easily check this, if you do any on-line banking. If you deposit money into your account and try to access the money the same evening, chances are that it will be unavailable and will remain so until the next day. This is because a batch job is run at night to update the master file with each deposit. Similarly, periodic functions such as billing or payments or course registration wait for a particular date and then process all the information on their files or databases at that point in time.

Some of these batch jobs run a long while. I have known jobs that run for 24 hours updating files.

**DBMS**

A database management system in the mainframe is embodied by IBM’s Information Management System (IMS). IMS is hierarchic in nature as opposed to the current relational databases. Its design was based mainly on the hierarchic organization of corporate structures. For quite some time, IMS was the only database system available in mainframe systems and was quite useful in developing some complex applications. It continues to be very important in big corporations, since many of their legacy applications are developed using IMS.

In the late 1980s, when relational database models became fashionable, IBM introduced DB2. After some initial hiccups, DB2 has become fairly common in many mainframe shops. Along with DB2 came the SQL (Structured Query Language). Both IMS databases and DB2 relational tables can be accessed from COBOL programs using their own internal APIs (application program interfaces). For IMS, the API is the DLI language, and DB2 has its own application program interface via the SQL.

When we talk of databases we must mention the Virtual Storage Data Access Method (VSAM). Both IMS and DB2 are implemented through VSAM. We can also define and access VSAM data sets, and in the mainframe environments VSAM has become a standard direct-access method.
Utility Programs

VSAM data sets can be created using a program from IBM called IDCAMS, and this has become pretty well the standard in all mainframe shops. Physical implementation of IMS databases is through VSAM data sets. IBM supplies a whole list of utilities for manipulating data sets; some of the standard ones are IEBGENER, IEBCOPY, IEFBR14, and IEBUPDT. These are utility programs, which can be run using JCL to manipulate data sets.

Some concepts in the mainframe come from the utilities associated with databases. For example, in order to define an IMS database we need what is called a Database Definition (DBD). In order to control access to an IMS database we need a Program Specification Block (PSB). A PSB consists of several Program Control Blocks (PCBs) for each DBD in the PSB. Developers use SQL Processor Using File Input (SPUFI) to manipulate DB2 tables.

A commonly used utility in the mainframe environment is the SORT utility. Large applications process voluminous amounts of data in the batch mode. Generally, these applications update some sort of a master file from transactions generated during the workday. This requires sorting the files in the proper order. Hence, most batch applications use the SORT function to sort their input and output files. SORT plays a very important role in the mainframe environment. Gigantic files can take a long time to sort, and the mainframe is very good at this type of processing.

OLTP

On-line applications are developed by programmers using IMS DC or CICS. These applications allow you to access data from IMS databases or DB2 tables in real time. IMS DC is a message-processing system and is very powerful. It provides a versatile environment where you can develop real-time transaction-processing systems. CICS, on the other hand, is much simpler and easier to work with while developing on-line applications. IMS transactions are processed in dedicated IMS regions, and DBAs (database administrators) generally manipulate the parameters associated with these regions. In general, IMS databases and DB2 tables can be immense. The mainframe is capable of processing these huge databases quite efficiently and in a timely manner. This is the primary reason why mainframes are used for large applications. Associated with IMS DC is the MFS (Message Formatting System); the corresponding item in the CICS environment is the BMS (Basic Map Support) definitions.
Reports

A typical computer application system has three phases: input, processing, and output. Generally output is in the form of reports. These reports can take various forms—for example, utility bills, statistical reports, or the paychecks you get paid with. Reports can be generated to provide valuable statistical information for the marketing people to analyze their data.

Report programs are pretty standard in the sense that they all have the same general features, but the details vary. COBOL can be used in the mainframe environment to generate reports. Several utilities also can be used. Some of these are the so-called fourth-generation languages. In the past it was MARK IV and RAMIS. Currently a lot of shops use EASYTRIEVE and SAS. One can even use some of the features of SYNCSORT to generate reports. EASYTRIEVE and SAS generally are easy to learn and can be used to generate quick ad-hoc reports. Complex reports, though, need more time in any language, whether it is EASYTRIEVE, SAS, SYNCSORT, or COBOL.

Releases and Testing

Source-code maintenance plays an important role in any development shop. Different shops have different methods. Most shops ship their software in periodic versions known as releases, and they need a way to maintain the different versions of their software. LIBRARIAN from Computer Associates is a product used to maintain source-code libraries. Source code can be compared using COMPAREX, a useful product from Compuware Inc. COMPAREX can be used to compare data sets to identify out-of-sync conditions.

ISPF has limitations in editing files with records longer than 255 bytes. To overcome these limitations we can use a product called File-AID. This extremely sophisticated tool is used to update files, or to map records with record maps or copy members. The batch part of File-AID can be used to selectively retrieve records from tapes or files, especially the large ones that are not easy to browse. File-AID and COMPAREX can be used in the testing phases of mainframe application systems to verify the results. File-AID has an IMS extension that can be used to browse, edit, and update IMS databases.

Many developers use a command language called CLIST to generate tools that are useful in application development. Another product, REXX, in some ways is better than CLIST in developing commands in the mainframe. ISPF Dialog Manager system can be used to develop panel-based application systems.

Many other vendor-supplied utilities are available in the mainframe environment. Some are performance related and are commonly used by systems programmers. Here we have discussed only the most common ones, which are used by most of the development shops.