LESSON 14:
SOFTWARE ARCHITECTURE

Objective

The objective here is to learn software architecture.

Introduction

Software Architecture

Software of the stored program control (SPC) systems arc of two types. Software of SPC systems is same as the general-purpose computer.

1. System Software
2. Application Software.

Software architecture deals with the system software environment including the language processors. We already know that O&M functions are usually carried out by general-purpose computer. Therefore, the system software architecture required for this purpose is similar to that of a general data processing system. Call processing is specific operation to switching systems. It demands real time responses and requires a software system with special features. Here we discuss the SPC software architecture only regarding to call processing functions.

Application software will be discussed in the next section.

Call processing is an event oriented processing function. It is triggered or turned on by event occurring at subscriber line or trunk. Call set up is not done in one continuous processing sequence in the telephone exchange. But, it involves several elementary processing actions. Each elementary processing action lasts after few tens or hundreds of milliseconds separated by periods of waiting for external events. Sometimes, these waiting periods may be as long as 20 m seconds. In a processor, many calls are processed simultaneously. Each call being handled by a separate process. A process, in this context, is a program in execution. A program by itself is not a process. In other words, we can say that n program is a passive entity, whereas process is active entity. In some books, terms task is used for process.

The multiple process environments are provided by multiprogramming feature. Now, we can note that in a 30,000-subscriber line exchange, there may be 3000 calls in the established state carrying speech transmission and another 500 in the state of being established or released. Thus, an important characteristic of system software of a switching processor is a powerful multiprogramming environment. This environment is capable of supporting as many as a few thousands of processors simultaneously.

Process in a multiprogramming environment may be in one of the following states:

- Running,
- Ready and
- Blocked state.

The state of a process is defined by its current activity. If the process executes then its state undergoes transitions. It is illustrated in Fig.14.1

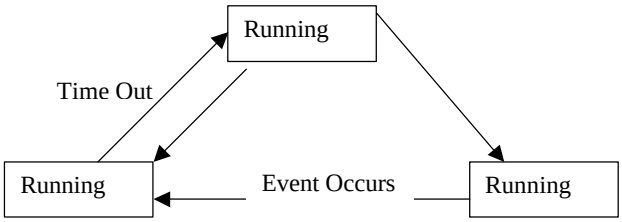

Fig. 14.1 Illustration of process states and transitions.

There are a large number of processes present in the system. But, CPU can be allocated to only one process at any instant of time.

Running Process: A process is said to be running if the CPU is allocated to it in current.

Ready Process: A process is said to he ready if it could use a CPU if one were available.

Blocked Process: A process is said to be blocked if it is waiting for some event to occur before it can at any time and several processes may be ready and several blocked.

The ready processes are ordered according to some priority. Therefore, the next process to receive the CPU is the first ready process in the ordered list. In the basis of level of priority, several ready lists can he prepared for each level of priority. The blocked processes are unordered and they unblock in the order in which the events they arc awaiting occur. A timer is set either accidentally or maliciously to prevent anyone process from monopolising the CPU. If the timer runs out, the process is forced to the ready state to join at the end of the appropriate ready list as prepared by its priority.

Each process is represented in operating system by a Process Control Block (PCB). It is a data structure containing, inter alia, the following information about the process:

1. Current state of the process
2. Process priority and CPU scheduling parameters.
3. Register save area to save the contents of the CPU registers when an interrupt occurs.
4. Memory allocated to the process.
5. Process accounts like the CPU time limits and usage, process number etc.
6. Status of events and I/O resources associated with the process.

Process Control Block (PCB) is a repository of all the key information about the processes required to restart a new process when it next gels the CPU. The CPU registers include a

program status word (PSW), the types of interrupts enabled or disabled currently, etc.

The PSW contain the address of the next instruction to be executed. Fig. 14.2 illustrates the switching of CPU between processors.

Process switching is also called context switching. If the currently running process becomes blocked or an event or interrupt triggers a high priority process then this process is called process switching.

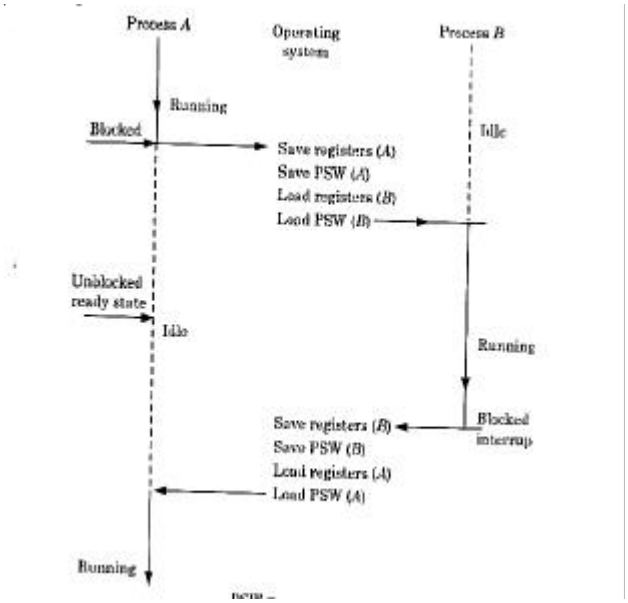

Fig. 14.2 Illustration of process switching

Typical priority levels in a telephone exchange in descending order of priority are given as follows:

1. Fault Alarms
2. Inter processor communication, high-speed peripherals such as desk etc.
3. High speed clock interrupt for periodic tasks.
4. Telephone exchange peripherals.
5. Low speed clock interrupt for periodic tasks.
6. Call processing.
7. Slow speed peripherals such as terminals etc.
8. Operation and Maintenance (O&M) tasks.

We know that context switching implies PCB saving and loading. Context switching occurs very often in a switching process. In order to speed up the context switching, we use special hardware instructions in switching processors for saving and loading PCBs.

Processes in 11 switching system cooperate with each other. They share common variables, update common tables, write into common files, and so on. Information about the resources of the telephone exchange such us trunks, registers etc. and their current utilisation is kept in the form of tables. For example, a state table of subscribed line has the information about each subscriber line whether it is free or busy.

Now we consider some process 'A'. This process scans the state table of subscriber line and finds that a particular subscriber is free. At the same instant of time, another process in which has higher priority interrupts and seeks the same subscriber line and sets it busy for its own purpose. When the control is returned to process 'A', it is blissfully unaware of the action of process '13' and sets the subscriber line busy once again and allocates the same for its own purpose. Thus, a single subscriber line is now allocated to two different processes (i.e. calls) at the same time, which is incorrect.

The similar problem can arise with other shared tables and files also. Such a problem can be solved by giving each process exclusive access to a shared table. When one process accesses a shared table, all others wanting to access the same table are kept waiting when accessing of the first process finishes, then one of the waiting processes is given access. Thus, there is mutual exclusion of processes in accessing shared data.

When a process is accessing shared data it is said to be in its critical section or critical region. Mutual exclusion implies that only one process can be in critical region at any instant of time for n given shared data. When one process is in its critical region, other processes may continue execution outside their critical regions. Since a process is in it critical region it prevents other processes entering this critical regions. The critical region of any process must execute as quickly as possible. A process cannot block within its critical region. The cooling of the critical sections or regions must be done carefully to avoid infinite loops etc.

There are many hardware and software solutions that can be devised to enforce mutual exclusion; here we shall discuss one generalised software solutions:

A synchronisation tool that is used here is called a Semaphore.

A semaphore is a protected variable that can assume non-negative integer values. We use one semaphore for each shared resource in the system. The initial value of a semaphore is set equal to the number of identical resources in a tool (say the number of trunk lines). There are two standard individual operations test (P) and increment (V) possible on a semaphore (S). Each P operation tests S to see that it is non-zero or not. If S is nonzero then there is a decrement in the value of S by one. It indicates that a resource has been removed from the common pool.

If S is zero, the process is blocked to be released by a V operation which increments the value of S by one signifying the return of a resource to the tool. A blocked process is placed in a queue of processes, which are blocked for want of the same resource. It can be assumed that S can be only binary values for the case of a shared table.

The implementation of a semaphore with a blocked queue may result in a situation where two or more processes are waiting indefinitely for a V operation. The V operation can be caused only by one of the blocked processes. If this happens, the processes arc said to be deadlocked. In order to illustrate above situation, we consider a system consisting of two processes P_0 and P_1 each process is accessing two binary semaphores S_0 and S_1

initiated to the value 1. The operations performed by processes P_0 and P_1 are:

Process P_0	Process P_1
P (S_0); $S_0 = S_0 - 1$	P (S_1); $S_1 = S_1 - 1$
P (S_1); blocked (S_1)	P (S_0); blocked (S_0)

The process Po having seized the semaphore S_0 gets blocked on S_1. The process P_1 having seized the semaphore S_1 gets blocked on S_0.

Both the processes Po and Pl are blocked for a V operation to be performed by the other and there is a deadlock. Techniques for deadlock prevention, avoidance, detection and recovery have been developed.

We have already discussed about salient features of operating system for switching processors. Now we shall discuss about the software production language processors. There are 5 basic factors associated with switching that is why it has important place in the switching industry:

1. Complexity and the size of the software.
2. Long working life required.
3. Real time operation is needed.
4. Stringent reliability and availability.
5. Software portability.

The complexity of switching software systems from the complexity of the switching systems.

Stored Program Control (SPC) System Software runs into hundreds of thousands of lines of code. The life expectancy of SPC software is about 10 years. It is quite exceptional compared with that of 10 or 15 years for software of other major systems.

Requirement of long life of switching software is the need to develop and expand it during its lifetime for the following purposes:

1. For supporting new services.
2. For meeting new requirements that can arise in network management.
3. For copying with applications that may differ from country to country.
4. For permitting technological enhancements in system hardware.

The high investment of cost and manpower required to develop switching software has led to the adaptation of software systems of successive generations of hardware. This is known as software probability.

Software engineering is a special branch of engineering where we study the problem's encountered in the production and maintenance of large-scale software for complex systems. The practice of software engineering techniques cause for four stages in the production of software systems:

1. Functional Specifications.
2. Formal Description and Detailed Specifications.
3. Loading and Verification.
4. Testing and Debugging.

The recommended approach to software design is top-down with an increasing level of complexity and abstractions. Here top-down means that proceeding from the general to the particular. To design a system, its functional specifications are usually described first in natural language. The formal description and specification language is usually based on diagrams. It clearly shows all possible imbrications and bifurcations that can occur in the program.

The next stage in software production is the programming itself. This is a translation process. This process converts the computer related part of the formal specification into programming language. Programs are partitioned into modules. Modules are developed using a stepwise decomposition or refinement process. Intermodule interfaces are carefully worked out so that modules lire relatively independent, making them easier to code, test and later modify. Program testing and validation is the last stage in software production. During this stage, errors in the program arc eliminated to a large extent by carefully selecting test patterns. This phase of activity is usually carried out in a bottom-up fashion. There are three major areas for software standardization. Two of them correspond to two stages in software production, viz. formal specifications and coding. Third is the language for man-machine interaction required for performing O&M functions in a telephone exchange.

Work in these areas has resulted in three CCIT Standards in the 'Z' series.

1. Z.100: This is a specification description language (SDL)
2. Z.200: CCITT high-level language (CHILL)
3. Z.3XX: Man-Machine Language (MML)

Now we shall discuss above language in subsequent subsections.

Z.100 - SDL

SDL is a formalised method of describing the functional Specifications and the internal processes necessary to implement the specifications. It is both a software development tool and a high-level documentation technique. This language is based primarily on state transition diagram. It provides traditional flowcharting features to represent sequential and conditional computations. It also has features termed signals. These are used to represent communication and synchronisation between concurrent processes. Single concurrent process is described by a single SDL diagram.

Various Standard Symbols of SDL are shown in Fig.14.3.

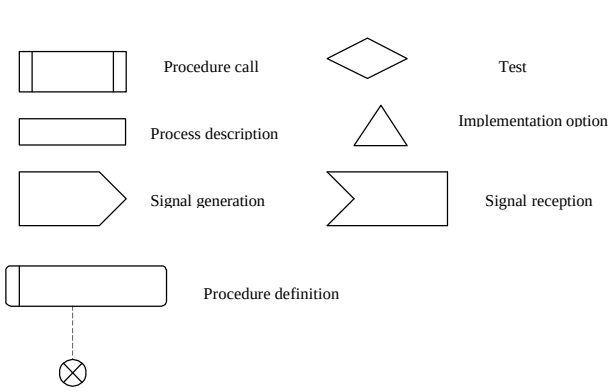

Fig.14.3 Various Standard symbols in SDL

Advantages offered by SOL to both switching equipment manufacturer and O&M agency of the system. It has following advantages:

1. It is a good top-down design technique.
2. It is oriented towards process interactions in a switching system.
3. Many tools are available for storage, updating and logical verification of SDL specifications. The SDL diagrams are easily transformed in Petri Nets. Petri Nets is an excellent mathematical tool for analysis and study of concurrent processes.
4. Many tools are available for translating SDL specifications into CHILL code and vice versa.
5. SDL is easy to learn, interpret and use.
6. It gives unambiguous specifications and descriptions for tendering und evaluation of offers.
7. It gives a basis for meaningful comparison of the capabilities of different SPC systems.

Switching systems basically belong to the class of finite state machines WSM). Here FSMs are asynchronous in nature and follow n sequential logic for their operation. Switching systems can he modelled by using a combinational part and a memory part. These are shown in Fig. 14.4.

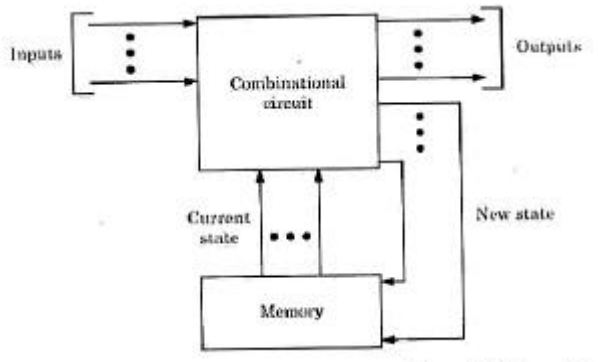

Fig. 14.4 Illustration of finite state machine (FSM) model.

In FSM, the status of the output circuits not only depends upon the inputs but also upon the current stale of the machine. Asynchronous sequential operation gives rise to many problems. These problems arise due to transient variations that may occur in the logic circuits and memory elements.

These problems can be overcome in clocked synchronous operation.

A clocked synchronous FSM is shown in Fig-. 14.5 the theory of the operating principles of synchronous FSM forms the basis of design of SDL.

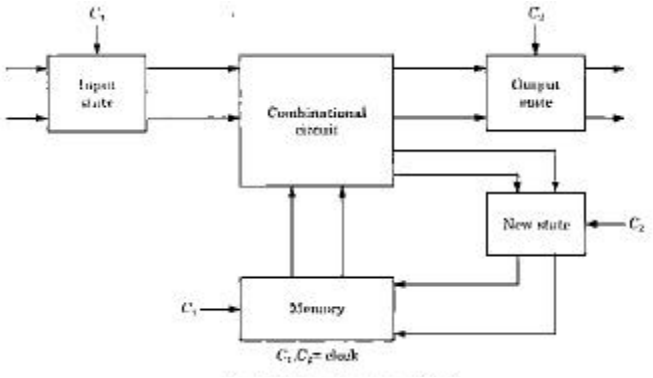

Fig.14.5 Synchronous FSM

Z.200, CHILL

Both assembly and high level languages are used in producing switching software. Assembly language programming was used in early electronic switching- systems. But, now-a-days trend is to use more and e of high level languages to the exclusion of assembly language. Due to the special processing requirements of a switching system, the high level languages such us PASCAL, ALGOL and FORTHAN are not ideally suited for developing SPC software. CCITT approved a high-level language for switching systems. It is recommended as Z.200. The language is known as CCITT high level language (CHILL).

It has been specially designed as suitable for coding the following SPC modules:

1. Call Handling
2. Test and Maintenance.
3. Operating system.
4. On-line and off-line support.
5. Man-Machine Language Translation.
6. Acceptance Testing.

CHILL has three major features like many other programming languages such as PASCAL:

1.Data Structure

2.Action Statements.

3.Program structure.

Data structure: CHILL supports both simple and complex data objects that are 'typed'.

Either a value or a location is called a simple data object. A location is referred to by a name and a value may be stored in it.

Data objects are 'typed' by attaching a mode to each one of them.

The mode of an object defines the set of values the object may assume, the access method of the object is a location, and the valid operations on the values. Examples of the standard modes are integer, boolean and character.

Complex data objects are built by aggregating simple objects. A complex object is composed of objects of the same mode (e.g. arrays) or objects of different mode. For objects of different modes, complex object is known as a structure. Each component object of a structure is called fuld.

Action Statements

These constitute the algorithmic part of CHILL program. CHILL supports assignment, procedure call and built in routine call. To control the program flow, CHILL provides the chemical control flow statements:

If for a two-way branch

CASE for a multi way branch,

DO for iteration and

CAUSE for specific exception.

Concurrent computation being a standard in switching software,CHILL has a provision for controlling concurrent computational flow. Some of the concurrent now control statements are START, STOP, DELAY, CONTINUE and RECEIVE EXPHESSION.

Program Structure

Program structuring consists of grouping together actions and objects that arc related. Program structuring controls the use of names (e.g. local or global) and the life-time of objects. For enabling concurrent computation, CHILL program structure supports process concept that we have discussed at the beginning of this section.

Various operations on processes are:

Create, destroy, suspend, resume, black, name, and set priority.

Z.3XX - Man-Machine Language (MML)

No switching system is conceivable without arrangements for Operation and maintenance (O&M) of the same. Electronic switching systems are no exception. CCITT man-machine language (MML) was initiated in early 1970s. The basics of (MML) were approved as recommendation Z.3xx series.

MML covers four main functions:

1. Operation
2. Maintenance
3. Installation
4. Acceptance Testing.

MML is designed to he used by novices as well as experts. It is to be adoptable to different national languages and organisations. It is to be flexible to allow incorporation of new technology. Today, MML is used at telephone exchange control desks as well as at O & M centres administering subscriber connections, routing, and performing traffic measurement and net-work management.

MML is defined by its syntax, semantics and information interchange procedures.

Syntax: It defines the character set and the use of symbols, keywords and special codes to construct grammatically correct language sequences. In MML, the syntax definitions are conveyed through syntax diagrams. These syntax diagrams in M11L arc similar to ones used in PASCAL.

Semantics: It defines the interpretation of each language element or statement.

Information interchange procedures:

These procedures deal with the conversational or interactive aspects of the man· machine dialogue required in the switching system operation and maintenance. An operator error can sometimes have serious consequences for the telephone exchange. MML information interchange procedures are defined to present such occurrences. Operators are provided access to the system only through functional commands like place a line in service, take a line out of service, modify service entitlement etc.

No direct access to data tables in the memory is permitted. For commands that have potentially serious consequences, MML procedure demands a reconfirmation from the operator before they are executed, there by drawing the attention of the operator to the risks attended on the requested action. l\IML procedures also have a roll back mechanism to resume operation from an easier state of the system in case the present comment is not properly executed due to a momentary failure in the system.