CHAPTER 4.

SEQUENCING

4.1 The Sequencing Set.

An event is an active phase of a process. When an event is scheduled, an event notice is generated referring to the process to become active. The event notice is included in the sequencing set, SQS, which is an ordered set of event notices. There can be at most one event notice referencing a given process.

The reference from an event notice to a process is through an element. I.e. the event notice refers to an element, which in turn references the process. The referenced element may or may not be a member of a set; its own SM has nothing to do with the SQS.

The SQS contains all event notices referring to events scheduled but not yet completed. Its first member is the event notice corresponding to the event currently in execution, the current event. The other events referred to in the SQS are called future events. The event whose event notice is the successor of the event notice of the current event is called the next event.

When an event is completed, its event notice is removed from the SQS and the next event becomes the current event.

The currently active process is referenced by the element expression "current". The element is the one referred to by the current event notice.

Statements operating on the SQS are called sequencing statements. The execution of a sequencing statement may terminate the current event. An event is terminated either by executing a sequencing statement, or by leaving the active process, through the final end of its operation rule, or by going to a label not local to the process.

A sequencing statement generating an event notice is called a scheduling statement. A scheduling statement assigns a real number to the event notice, which determines its position in the SQS. Because of its use in system descriptions the number is called the time reference of the event notice. The time reference of an individual event notice remains fixed.

The time reference of the current event is called the current system time. It is accessed by the real procedure "time". The SQS can be regarded as a representation of the "system time axis". The fact that the system time remains constant during an active phase of a process, shows that the execution of a SIMULA program is a sequence of discrete events, when viewed in system time.

A scheduling statement will insert the generated event notice in the SQS behind all event notices with a smaller time reference and in front of all events with a greater time reference. The new event notice normally goes in behind those with the same time reference, if any; but the scheduling statement can specify insertion with priority, in which case the event notice is placed in front of those with the same time reference. Its position can also be explicitly stated.

If an event notice with the time reference equal to the current system time is inserted with priority, it takes precedence over the current event notice. The current event becomes the next event (a reactivation point is defined for the process), and the scheduled event becomes the current event. The effect of such a scheduling statement is to some extent similar to that of a procedure call.

4.2 Sequencing Statements.

The following sequencing statements may operate on the SQS by removing an event notice. If it is the current event notice, the current active phase is terminated. The state of the referenced process is changed accordingly. The element through which the process is referenced need not be the same as the one referred to by the event notice.

1. terminate(X).

   The process referred to by X becomes terminated. Any event notice referring to the process is removed from the SQS, and any reactivation point is deleted. The effect is as if control had left the process through its final end. If X has no process aspect the statement has no effect.

With the exception of this statement any sequencing statement, including those discussed in the following section, will define the reactivation point of the currently active process, if the current active phase is terminated. The reactivation point is on the next statement in the dynamic sense.

2. cancel(X).

   If the process referred to by X is active or suspended, the associated event notice is removed from the SQS, and the process becomes passive. A reactivation point is defined if the process was active, and is left unchanged if it was suspended. In all other cases the statement has no effect.

3. passivate.

   This statement is equivalent to cancel(current).

4. wait(S).

   This statement is equivalent to the compound statement begin include(current, S); passivate end.

4.3 Scheduling Statements.

The following sequencing statements may schedule an event, and some may in addition cancel an event already in the SQS.

The statement

hold(T)

where T is an arithmetic expression with a non-negative value, will cancel the current event, terminating the current active phase, and schedule a new event for the same process. The event notice gets the time reference "time + T", and is included in the SQS without priority. Viewed in system time the statement normally represents a suspension period of length T for the current process, after which control proceeds to the next statement in the dynamic sense. Notice, however, that the event scheduled for the current process may be cancelled by another process before it is executed.

Since the future event is inserted without priority, the statement hold(0) will allow further events with the same time reference as the current one to be executed, after which the current process resumes control. If T has a negative value, hold(T) is equivalent to hold(0).

With the exception of the "hold" statement all scheduling statements follow a special syntax which allows greater flexibility than do ordinary procedure statements. The general format is
<activation clause> <scheduling clause>
in which the latter clause is optional. Connection clauses may be added (see CHAPTER 5).

A typical scheduling statement is

activate X at T,

where X is an element expression and T is an arithmetic expression. The statement has an effect if and only if X refers to a passive process. Viewed in system time the next active phase of that process is scheduled for the time T.

An event notice is generated with the time reference equal to the greater of T and the current system time. It is inserted in the SQS without priority. The event notice refers to the element which is the current value of the expression X. This element becomes the value of the function "current" during the execution of the scheduled active phase. Notice that the SM of the element (but not the PA may have been altered in the meantime).

Another option is to write

activate X delay T;

where the time reference is specified relative to the current system time. The scheduling clause "delay T" is equivalent to "at time + T".

Scheduling with priority is by writing

activate X at T prior; or activate X delay T prior;

The statement

activate X delay 0 prior; (or at time prior)

has the equivalent abbreviated form

activate X;

The event is scheduled in front of the current event, so that the indicated active phase is executed immediately in real time. The current active phase is terminated, but its event notice will remain in the SQS as the next event. Provided that this event is not cancelled, the indicated active phase functions as a "subroutine" to the current process. This is called direct scheduling.

The statements

activate X after Y ; and activate X before Y ;

have an effect only if the process referenced by Y is active or suspended, i.e. if there is an event notice referencing the process (not necessarily through the same element). The event notice generated for X is included in the SQS immediately after, or immediately before, the specified event notice. It is given the same time reference as the latter. The priority concept does not apply.

activate X before current;

is another way of specifying direct scheduling.

No activate statement has an effect if the process designated by X is active or suspended, i.e. if it is already represented in the SQS. However, similar statements with "activate" replaced by "reactivate" will have an effect in this case as well. The old event notice, if any, referring to the designated process is removed from the SQS, and then a new event is (or may be) scheduled as above.

If the designated process is not the currently active one, the statement

reactivate X at T;

has the same effect as

cancel(X); activate X at T;

(provided that the expression X evaluates to the same element both times and has no side effect).

The statement

reactivate current delay T;

is equivalent to hold(T).

If X refers to the currently active process (same(X,current) is true), the statement

reactivate X;

has no other visible effect than possibly changing the element identity (and set membership aspect) of "current".

The statement

reactivate X before Y; (or after Y)

has the same effect as cancel(X), if Y does not refer to an active or suspended process.

No scheduling statement will have an effect if the element X has no process aspect, or if the designated process is terminated.

4.4 SQS Functions.

The following function procedures are based on the SQS concept.

Element procedures.

1. current.

   The value is the element referred to by the current event notice. This element references the currently active process, i.e. the process in which the expression is evaluated.

2. Activity identifier.

   An activity identifier referenced within the corresponding activity declaration is a function designator equivalent to "current" except when preceded by the symbol "new" (see section 3.4.1) or the symbol "when" (see CHAPTER 5). When referenced within the body of a procedure declared local to the activity, it may occasionally assume a different significance (see CHAPTER 5).

3. nextev(X).

   If the PA of X is scheduled as an event in the SQS, i.e. is an active or suspended process, the function value refers to the process scheduled as the following event. The element is the one referenced by the corresponding event notice. The value is none if there is no following event in the SQS, or if the PA of X is passive or terminated, or if X has no PA.

Real procedures.

1. evtime(X).

   The value is equal to the time reference of the event notice corresponding to the referenced process. If the process is passive or terminated, or if X has no PA, the value is undefined.

2. time.

   The value is equal to evtime(current).

Boolean procedures.

1. idle(X).

   The value is true if X refers to a passive or terminated process, or if X has no PA.

2. finished(X).

   The value is true if X refers to a terminated process, or if X has no PA.

Examples.

The state of a process can be assessed by the following Boolean expressions:

active:	same(X, current),
suspended:	¬ idle(X)∧(¬ same(X,current)),
passive:	idle(X)∧(¬ finished(X)), and
terminated:	finished(X)

4.5 Examples.

1. Postponement.

   reactivate X at evtime(X) + T;

   Provided that the referenced process is suspended the statement will increase its suspension period by T.

2. Scheduling for an unknown system time.

   If X refers to a suspended process the statement

   reactivate current after X;

   will invoke a suspension period of the currently active process such that its next active phase comes immediately after that of X (provided that neither of these events is cancelled before execution).

   A similar effect can be obtained by means of the following

procedure, if the PA of X may be a passive process. It is assumed that X is an element variable, and that all relevant activate statements in the system refer to the process via this variable.

```
procedure link(Z); element Z;
if idle(Z) then begin element swap;
          swap: = Z; Z: = current; passivate;
          Z: = swap; activate Z end
else reactivate current after Z;
```

Notice that the element "current" refers to the process containing the call for the link procedure currently in execution. The passivate statement also pertains to that process. It follows that the statement

```
link(X)
```

invokes an inactive period (passive or suspended) of the currently active process of the required length.

3. Debugging.

A process of the following class will "screen" the events executed in the system, except events invoked by direct scheduling. Special actions can be taken before and afte active phases of the process specified by the exogenous element attribute X, e.g. debugging information can be printed.

```
activity monitor(X); element X;
begin element Y;
next: Y: = nextev(current);
rep: if similar(Y,current) then
          begin Y: = nextev(Y); go to rep end;
     if same (X,Y) then
          begin reactivate current before Y;
```

```
                    actions1; reactivate Y; actions2 end
            else reactivate current after Y;
            go to next end;
```

The short loop beginning at label "rep" makes it possible for any number of monitor processes to operate in parallel. Any number of processes can therefore be monitored.

The monitor processes will not interfere with the operation of the system, provided that there is no reference to the procedure "nextev".