Published: 06/16/67

Identification

Process Id Generator
J. J. Donovan

Purpose

Process id generator is a procedure in Process Control. Process id returns a unique identifier that may be assigned to a newly created process.

Discussion

The purpose of process id generator is to return to its caller a unique identifier. We have imposed the following constraints on the unique identifier. These constraints are arbitrary and were chosen to facilitate the implementation of process id in Multics.

The first constraint is that the unique identifier produced by process id is a 36 bit word. A 36 bit word was chosen because the read-alter-rewrite system controller interface for the GE.600 series operates on a 36 bit word. Instructions based on this interface are used for interlocking data, and the process id makes a good interlock word.

A 36 bit reading of the 52 bit calendar clock will be used to initialize the series of unique identifiers produced by process id generator. A clock reading was chosen because it is sequential and if the system should go down merely taking a new clock reading will result in initializing a series of ids which have not been used before.

The problem with using a 52 bit clock reading is that the 52 bit reading must be reduced to 36 bits to initialize the series of process ids. Process id generator does this reduction by masking. However, a trade off results. Masking off the high order bits of the clock results in a period, a number of hours, after which the 36 bit configuration will repeat. Masking of the low order bits results in a lower limit, a minimum time interval, in which time all 36 bit configurations are the same. We have made a compromise. Five of the high order bits and eleven of the low order bits will be masked. The implication of this choice of masking is that we are assuming that the life time of a process in Multics is not over four years, and that the system will not be creating processes faster than every two milliseconds. (See a future section on the clock for a table of alternate compromises.)

Implementation

Process id generator will be called by the following:

```
call get_proc_id (id);
```

In this call the argument id is the returned unique id. The PL/I declaration of the parameter used in the call is:

```
dcl id bit (36);
```

Process id will have a system-wide data base consisting of two words, an interlock word and a data word containing the last process id created. Process id must lock its data base while using it.

Process id is initialized by reading the clock. Process id, written in EPLBSA, will use an

```
rccl <clock>|0,*
```

to read clock. The clock reading will appear in the AQ right justified. Process id constructs a 36 bit word from this 72 bit configuration by masking off the first 25 bits and the last 11 bits.

After initializing the first process id subsequent ids will be created by simply adding one to the value of the last id created.

Process id will compare the most recent id created to the present clock reading. If they are equal process id will repeat the test over and over until the clock advances enough that the test succeeds. If the value of the id is less than the value of the clock the value of the id will be returned to the caller of process id. In this manner process id will ensure that the process id is never a larger value than the clock. If the system goes down a new series of process ids is initialized by reading the clock. Members of this new series of process id's will be different from any old process id since the process ids have never gotten ahead of the clock. To initialize the series of process ids, the entry point

```
call get_proc_id$init;
```

is called.