on return from that procedure. It is the responsibility of callable procedures to check their pointer parameters to ensure that they don't have any addressing exceptions, and that they return values only to the user environment. A bug in the procedure `setlooptimer`, which sets a watchdog timer and optionally returns the old value, made it possible to become the `SUPER.SUPER` (the root user, with ID 255,255, or –1):

```
proc make^me^super main;
begin
int .TOS = 'S';                             -- top of stack address

call setlooptimer (%2017);                  -- set a timer value
call setlooptimer (0, @TOS [4]);            -- reset, return old value to saved E reg
pcb [mypid.<8:15>].pcbprocaid := -1;        -- dick in my PCB and make me super
end;
```

The second call to `setlooptimer` returns the old value `%2017` to the saved E register contents on stack, in particular setting the `priv` bit, which leaves the process in privileged state. Theoretically this value could have been decremented to `%2016`, but this would not make any difference (this is the saved RP field, which is not restored). The program then uses SG-relative addressing to modify the user information in its own process control block (PCB). `mypid` is a function returning the current process's PID, and the last 8 bits (`<8:15>`) were the PIN, which is used as an index in the PCB table.

This bug was quickly fixed, of course, but it showed a weakness in the approach: it is up to the programmer to check the parameters passed to callable procedures. Throughout the life of the architecture, such problems have reoccurred.

File Access Security

Tandem's approach to file access security is similar to that of Unix, but users can belong only to a single group, which is part of the username. Thus my username, `SUPPORT.GREG`, also written numerically as `20,102`, indicates that I belong to the `SUPPORT` group (20) only, and that within that group my user ID is 102. Each of these fields is 8 bits long, so the complete user ID fits in a word. If I wanted to be a member of another group, I would need another user ID, possibly with a different number—for example, `SUPER.GREG` with user ID 255,17.

Each file has a number of bits describing what access the owner, the group, or all users have to the file. Unlike Unix, however, the bits are organized differently: the four permissions are *read*, *write*, *execute*, and *purge*. *Purge* is the Tandem name for *delete*, and it's necessary because directories don't have their own security settings.

For each of these access modes, there is a choice of who is allowed to use them:

- *Owner* means only the owner of the file.
- *Group* means anybody in the same group.
- *All* means anybody.