Introducción a Sistemas Operativos: Concurrencia

Clips xxx
Francisco J Ballesteros

1. Deadlocks

Rara vez necesitaremos un único cierre. O, dicho de otro modo, rara vez necesitaremos un único recurso compartido. Lo normal es compartir diversos recursos. En principio cada recurso puede utilizar un cierre distinto y los procesos que necesitan acceder al recurso pueden adquirir su cierre durante la región crítica. Pero piensa lo que sucede si un proceso ejecuta

```
lock(a);
lock(b);
...
```

y otro en cambio ejecuta

```
lock(b);
lock(a);
```

Si ejecutamos el primer `lock` de cada proceso, ninguno podrá continuar jamás. Cada uno tiene un cierre y necesita el que tiene el otro. El efecto neto es que los procesos se quedan bloqueados de por vida y no terminan. A esto lo denominamos **deadlock** o **interbloqueo**. ¿Recuerdas qué sucedía si un proceso deja abierto un descriptor de escritura en un pipe y se pone a leer del mismo? Efectivamente, es un deadlock.

En el caso de cierres, para evitar el problema suele bastar fijar un **orden** a la hora de adquirir los recursos. Si todos adquieren los cierres que necesitan siguiendo el orden acordado, no tenemos interbloqueos. Por ejemplo, podemos acordar que el cierre de `a` ha de tomarse antes que el de `b` y el de `b` antes que el de `c`. De ser así, podríamos tener código para un proceso que ejecute

```
lock(a);
lock(c);
...
```

otro que ejecute

```
lock(a);
lock(b);
```

y otro

```
lock(b);
lock(c);
```

y en ningún caso tendríamos un bloqueo. Si alguien necesita un cierre nadie puede tenerlo y además estar esperando los que ya tenemos nosotros: no hay deadlocks.

Esto puede resultar complicado y en ocasiones se opta por usar un único cierre para todos los recursos, a lo que se suele denominar *giant lock* o "cierre gordo". Pero naturalmente se limita la concurrencia y se reduce el rendimiento. No obstante, hay menos posibilidades de tener condiciones de carrera por olvidar echar un cierre en el momento adecuado.

Hay veces en que se opta por dejar que suceda el deadlock y, tras detectarlo o comprobar que estamos ante un posible deadlock, soltar todos los cierres y volverlo a intentar. Esto no es realmente una solución dado que podríamos volver a caer en el interbloqueo una vez tras otra (aunque no es 100% seguro que siempre suceda tal cosa). A este tipo de bloqueos (que podrían romperse con cierta probabilidad) se los denomina **livelock**.

Lo que es más, puede que algún proceso tenga mala suerte y, si aplica esta estrategia, nunca consiga los cierres que necesita porque siempre gane otro proceso al obtener uno de los cierres que necesita. El pobre proceso continuaría sin poder trabajar, a lo que se denomina **starvation**, o **hambruna**.