INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

upiita-ipn

ALGORITMOS DE DEKKER Y PETERSON

PROFESORA:

Susana Sanchez Najera

INTEGRANTES DEL EQUIPO:

Vargas Sánchez Andrea Liliana
Cisneros Martínez Daphne Liliana
Escala Acosta Andres Rafael

GRUPO: 2TV3

Fecha de entrega: 15 de Septiembre del 2021

ALGORITMO DE DEKKER

Pone en práctica la exclusión mutua en el contexto de un programa concurrente con dos procesos: *parbegin / parend* hacen que ambos procesos sean procesos concurrentes, estos se encuentran en ciclos infinitos entrando en secciones críticas varias veces.

Entrar exclusión mutua se implementa con un ciclo while que se repite hasta que el numero_proceso sea igual al número del proceso.

En esta primera solución había solamente una variable global lo que ocasiona el problema de sincronización en tándem

```
program versión_uno
var número_proceso: integer ;
procedure proceso_uno
    begin
        while true do
            begin
                while número_proceso = 2 do;
                sección_crítica_uno;
                número_proceso := 2;
                otras_tareas_dos
            end
    end;
procedure proceso_dos;
    begin
        while true do
            begin
                while número_proceso = 1 do;
                sección_crítica_dos;
                número_proceso := 1;
                proceso_dos_proceso
            end
    end;
begin
    número_proceso := 1;
    parbegin
        proceso_uno;
        proceso_dos
    parend
end.
```

Fig. 4.3 Primera versión de las primitivas de exclusión mutua.

Imagen 1. Primera versión algoritmo de Dekker

En la versión dos para solucionar el problema se usa una segunda variable. De modo que ahora un proceso permanece bloqueado en espera activa mientras el otro proceso está adentro.

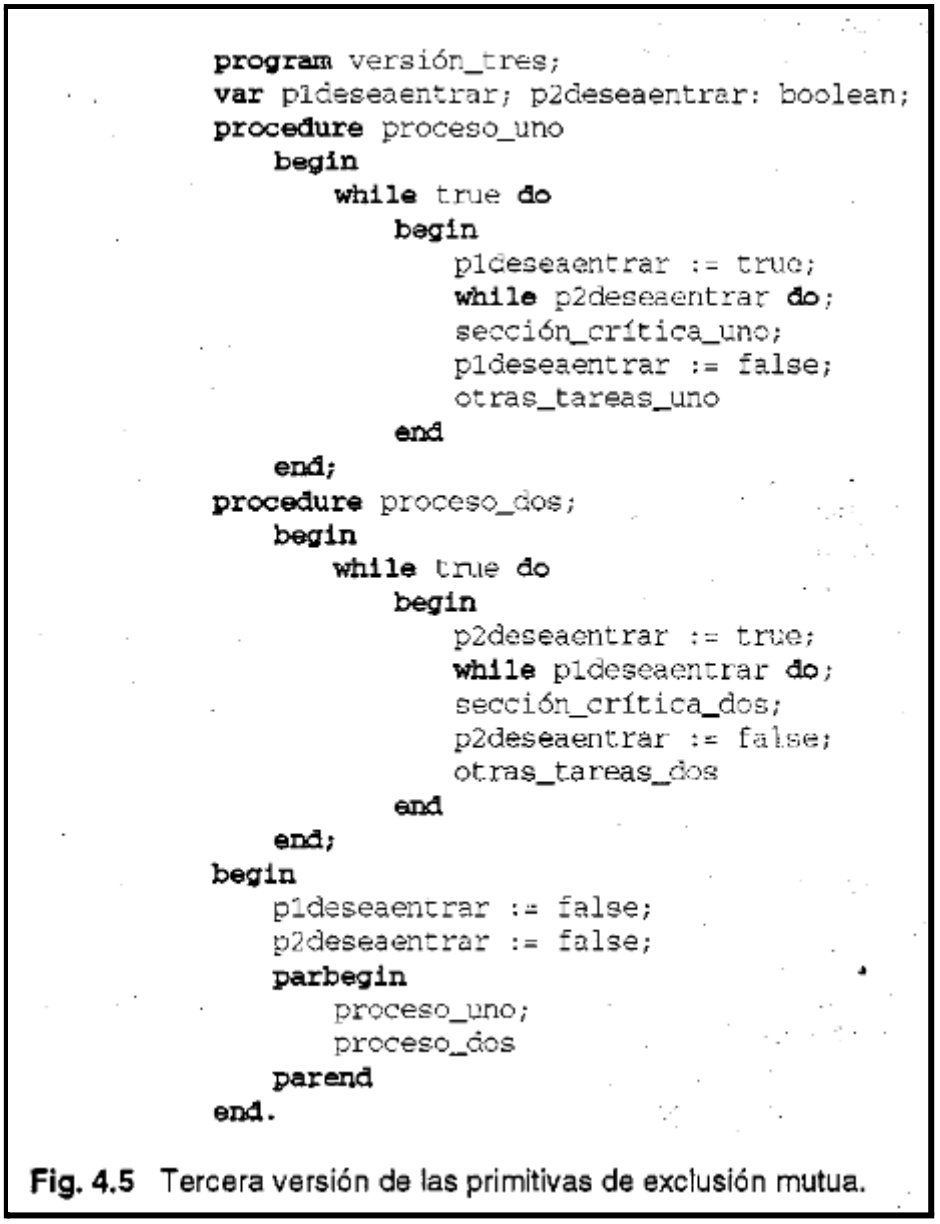
El problema en esta versión es que en lo que se hace la validación con el while hay tiempo suficiente para que otro proceso verifique su bandera y entre en su sección crítica.

```
program versión_dos
var p1adentro, p2adentro: boolean;
procedure proceso_uno;
    begin
        while true do
            begin
                while p2adentro do;
                p1adentro := true;
                sección_crítica_uno;
                p1adentro := false;
                otras_tareas_uno
            end
    end;
procedure proceso_dos;
    begin
        while true do;
            begin
                while p1adentro do;
                p2adentro := true;
                sección_crítica_dos;
                p2adentro := false;
                otras_tareas_dos
            end
    end;
begin
    p1adentro := false;
    p2adentro := false;
    parbegin
        proceso_uno;
        proceso_dos
    parend
end.
```

Fig. 4.4 Segunda version de las primitivas de exclusión mutua.

Imagen 2. Segunda versión algoritmo de Dekker

Este problema se resuelve en la versión 3 al asignar un valor de cierto a la bandera antes de realizar la verificación. De esta manera se obtiene una exclusión mutua. Pero se genera un problema nuevo, pues con los 2 en cierto encuentran la bandera del otro y entran en un ciclo infinito. Este es un ejemplo de bloqueo mutuo de dos procesos.

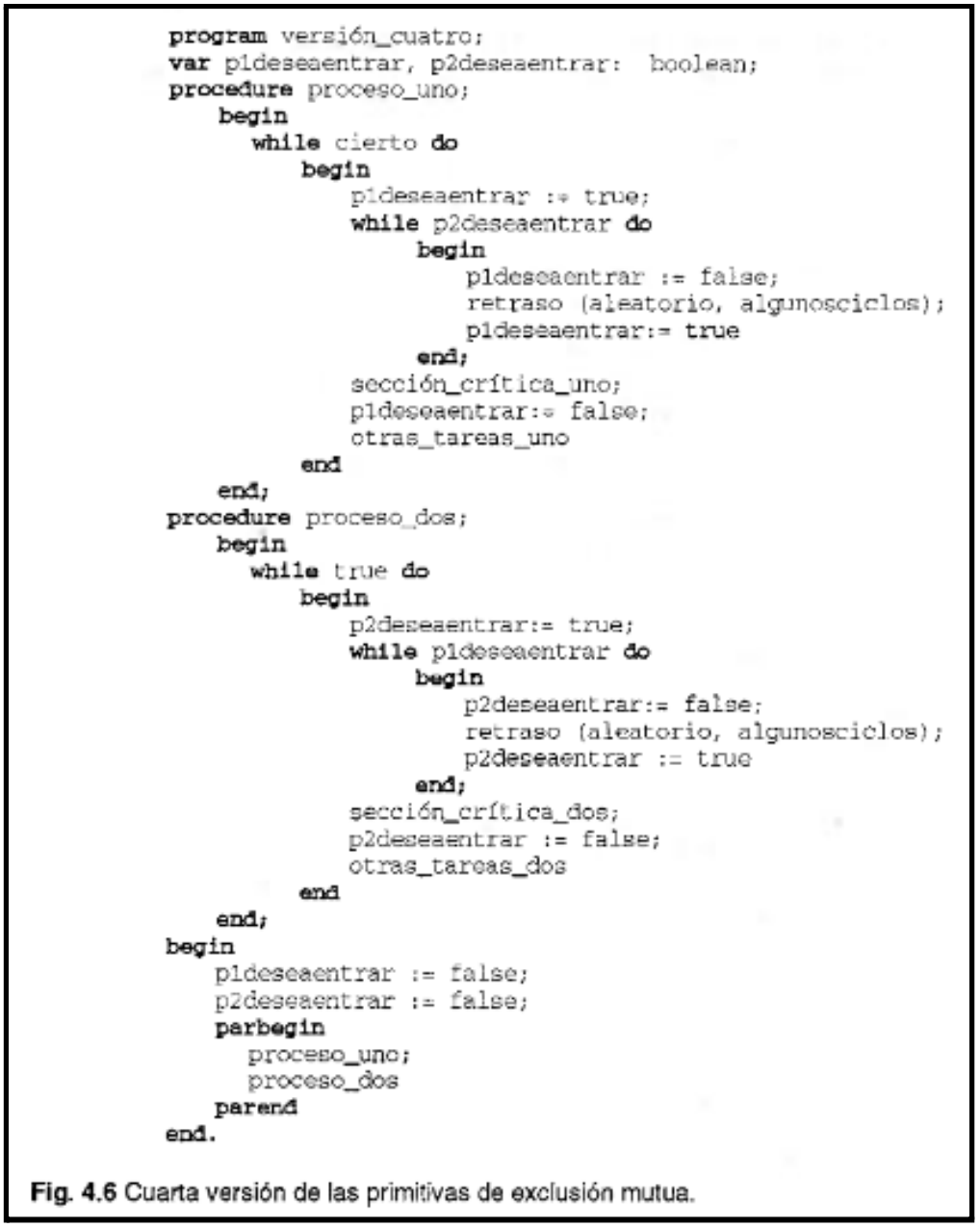


```
program versión_tres;
var p1deseaentrar; p2deseaentrar: boolean;
procedure proceso_uno
    begin
        while true do
            begin
                p1deseaentrar := true;
                while p2deseaentrar do;
                sección_crítica_uno;
                p1deseaentrar := false;
                otras_tareas_uno
            end
    end;
procedure proceso_dos;
    begin
        while true do
            begin
                p2deseaentrar := true;
                while p1deseaentrar do;
                sección_crítica_dos;
                p2deseaentrar := false;
                otras_tareas_dos
            end
    end;
begin
    p1deseaentrar := false;
    p2deseaentrar := false;
    parbegin
        proceso_uno;
        proceso_dos
    parend
end.
```

Fig. 4.5 Tercera versión de las primitivas de exclusión mutua.

Imagen 3. Tercera versión algoritmo de Dekker

En la versión 4 se necesita una forma de escapar de estas verificaciones, esto se obtiene ligando repetidamente a la bandera de proceso que se encuentre dentro del ciclo el valor de falso por periodos cortos.
Pero puede crear un nuevo problema que es un aplazamiento indefinido. Por lo que esta versión resulta obsoleta.



```
program versión_cuatro;
var p1deseaentrar, p2deseaentrar:  boolean;
procedure proceso_uno;
    begin
      while cierto do
          begin
              p1deseaentrar := true;
              while p2deseaentrar do
                   begin
                       p1deseaentrar := false;
                       retraso (aleatorio, algunosciclos);
                       p1deseaentrar:= true
                   end;
              sección_crítica_uno;
              p1deseaentrar:= false;
              otras_tareas_uno
          end
    end;
procedure proceso_dos;
    begin
      while true do
          begin
              p2deseaentrar:= true;
              while p1deseaentrar do
                   begin
                       p2deseaentrar:= false;
                       retraso (aleatorio, algunosciclos);
                       p2deseaentrar := true
                   end;
              sección_crítica_dos;
              p2deseaentrar := false;
              otras_tareas_dos
          end
    end;
begin
    p1deseaentrar := false;
    p2deseaentrar := false;
    parbegin
      proceso_uno;
      proceso_dos
    parend
end.
```

Fig. 4.6 Cuarta versión de las primitivas de exclusión mutua.

Imagen 4. Cuarta versión algoritmo de Dekker

El problema de esta versión se resuelve con el siguiente código y este es al que conocemos hoy en día como algoritmo de Dekker siendo el primer algoritmo funcional en conseguir la exclusión mutua de procesos.

```
program algoritmo_dekker;
var proceso_favorecido: (primero, segundo);
    p1deseaentrar, p2deseaentrar: boolean;
procedure proceso_uno;
    begin
      while true do
          begin
              p1deseaentrar := true;
              while p2deseaentrar do
                  if proceso_favorecido = segundo then
                      begin
                          p1deseaentrar := false;
                          while proceso_favorecido = segundo do;
                          p1deseaentrar := true
                      end;
                  sección_crítica_uno;
                  proceso_favorecido := segundo;
                  p1deseaentrar := false;
                  otras_tareas_uno
              end
    end;
procedure proceso_dos;
    begin
      while true do
          begin
              p2deseaentrar := true;
              while p1deseaentrar do
                  if proceso_favorecido = primero then
                      begin
                          p2deseaentrar := false;
                          while proceso_favorecido = primero do;
                          p2deseaentrar := true
                      end;
                  sección_crítica_dos;
                  proceso_favorecido := primero;
                  p2deseaentrar := false;
                  otras_tareas_dos
          end
    end;
begin
    p1deseaentrar := false;
    p2deseaentrar := false;
    proceso_favorecido := primero;
    parbegin
      proceso_uno;
      proceso_dos
      parend
end.
```

Fig. 4.7 Algoritmo de Dekker para realizar las primitivas de exclusión mutua.

Imagen 5. Algoritmo de Dekker con solución de problemas

ALGORITMO DE PETERSON

Este algoritmo es mucho más fácil de manejar para el caso de la exclusión mutua de dos procesos con espera activa.

El algoritmo se puede ver de la siguiente manera:

```
program algoritmo_peterson;
var proceso_favorecido: (primero, segundo);
      p1deseaentrar, p2deseaentrar: boolean;
procedure proceso_uno;
      begin
        while true do
          begin
            p1deseaentrar := true;
            proceso_favorecido := segundo;
            while p2deseaentrar
              and proceso_favorecido = segundo do;
            sección_crítica_uno;
            p1deseaentrar := false;
            otras_tareas_uno
          end
      end;
procedure proceso_dos;
      begin
        while true do
          begin
            p2deseaentrar := true;
            proceso_favorecido := primero;
            while p1deseaentrar
              and proceso_favorecido = primero do;
            sección_crítica_dos := false;
            otras_tareas_dos
          end
      end;
begin
      p1deseaentrar := false;
      p2deseaentrar := false;
      proceso_favorecido := primero;
      parbegin
        proceso_uno;
        proceso_dos
      parend
end.
```

Fig. 4.8 Algoritmo de Peterson para realizar las primitivas de exclusión mutua.

Imagen 6. Algoritmo de Peterson para exclusión mutua de procesos

CONCLUSIONES

Los procesos concurrentes pueden funcionar en forma totalmente independiente unos de otros, o pueden ser asíncronos, lo cual significa que en ocasiones requieren cierta sincronización y cooperación. Existen muchos problemas importantes de asincronía. Y, para resolver los problemas de la exclusión mutua y de sistemas de procesamiento paralelo, se desarrollaron los algoritmos de Dekker y de Peterson.

El algoritmo de Dekker se desarrolló para solucionar el problema del aplazamiento indefinido que surge en la cuarta versión de las primitivas de exclusión mutua. Y el algoritmo de Peterson es una simplificación y una mejora del algoritmo de Dekker, para manejar, de manera más sencilla, una exclusión mutua de dos procesos con espera activa

REFERENCIAS

H. Deitel. SISTEMAS OPERATIVOS. Unix OS/2 MS-DOS. Estados Unidos, Pearson, 2da Edición.