Ejercicios Concurrencia: Exclusión mútua y sincronización

7.

```
Pasajeros(){
    down(entrar_taxi);
        while(inicio_embarque);
    up(entrar_taxi);
    down(mutex);
    esperando++;
    up(mutex);
    down(espera);
    subir_avion();
    up(subido);
}
```

```
sem entrar_taxi = 1; // U n mutex
int inicio_embarque = 0; // si pueden embarcar
sem mutex = 1;
int esperando = 0; // Nº de pasajeros esperando
sem espera = 0; // espera al up del avión
sem subido = 0; // para esperar a la subida de los pasajeros
sem terminal = 1; // 1 terminal hoy
```

```
Avion(){
    down(terminal); inicio_embarque = 1;
    preparando_avion();
    down(mutex);
    n = esperando;
    up(mutex);
    up(espera, n);
    down(subido, n); up(mutex); esperando -= n; down(mutex);
    inicio_embarque = 0; despegue();
    up(terminal);
}
```

8.

```
Anestesista(){
    sedar_paciente();
    up(enfermera);
    down(anestesista);
    despertar();
}
```

```
sem enfermera = 0;
sem anestesista = 0;
sem instrumentos = 0;
sem gasas = 0;
sem cirujano = 0;
sem ocio = 0;
```

```
Enfermera(){
    down(enfermera);
    preparar_material();
    up(instrumentos, 3);
    up(gasas, 2);
    down(cirujano);
    down(instrumentos, 2);
    down(gasas, 1);
    up(instrumentos, 2);
    up(instrumentos, 1);
    down(instrumentos, 2);
    down(gasas, 2);
    down(ocio);
    up(cirujano);
```

```
Cirujano(){
    down(instrumentos, 3);
    down(gasas, 2);
    up(instrumentos, 2);
    up(gasas, 1);
    down(instrumentos, 1);
    down(gasas, 1);
    up(instrumentos, 2);
    up(gasas, 2);
    cose();
    up(ocio);
}
```

Scanned with CamScanner

Junco de los Heros
Valenzuela

10.)

```
Ciudadano(){
  down(mutex);
  ++nciudadano;
  up(mutex);
  down(espera);
  realizar_gestion();
}
```

```
Agente(){
  while(1){
    alarm(15*60);
    pensar();
    down(mutex);
    n = nciudadano;
    up(mutex);
    up(espera, n);
  }
}
```

```
sem mutex = 1;
int nciudadano = 0; // nº de ciudadanos esperando
sem espera = 0;
```

11.)

```
Estudiante(){
  down(mutexsilla);
  if(S==0){
    up(mutexsilla);
    return;
  }
  S--;
  up(mutexsilla);
  down(bandeja);
  up(lleno);
  down(comida);
  come_comida();
  up(bandeja);
  down(mutexsilla);
  S++;
  up(mutexsilla);
}
```

```
Cocinero(){
  while(1){
    down(lleno);
    sirve_comida();
    up(comida);
  }
}
```

```
sem mutexsilla = 1;
sem bandeja = B;
sem lleno = 0;
sem comida = 0;
} Reservas
```

17.) 1) Se necesitan al menos 2. Con 1 no se puede ya que un proceso hará wait y el otro up, y el de up puede no esperar, y con 2 sí que se puede (Rendezvous):

```
mutex1 = 0
mutex2 = 0
```

```
P1(){
  down(mutex1);
  up(mutex2);
}
```

```
P2(){
  up(mutex1);
  down(mutex2);
}
```

2) Para ordenar 3 procesos se necesitará 2·3 = 6 semáforos, ordenando 2 a 2 (P_1 con P_2, P_2 con P_3, P_1 con P_3) ya que si se omite uno, puede que el proceso que haga up libere un down que no debería.

Scanned with CamScanner

Juan de las Heras
Valenzuela

18.)

```
Proceso P1(){
  código P1
  up(finP1);
  up(finP1);
  exit(EXIT_SUCCESS);
}

Proceso P2(){
  down(finP1);
  código P2;
  up(finP2);
  up(finP2);
  exit(EXIT_SUCCESS);
}

Proceso P3(){
  down(finP1);
  código P3;
  up(fin P3);
  up(fin P3);
  exit(EXIT_SUCCESS);
}

Proceso P4(){
  down(finP2);
  código P4;
  up(finP4);
  exit(EXIT_SUCCESS);
}

Proceso P5(){
  down(finP2);
  down(fin P3);
  código P5;
  exit(EXIT_SUCCESS);
}

Proceso P6(){
  down(finP4);
  down(finP3);
  código P6;
  exit(EXIT_SUCCESS);
}
```

19.)

```
Guardia forestal(){
  down(guardia);
  up(inicio);
  down(billetes,10);
  for(int i=1; i<=10; i++){
    down(mutexinspector);
    if(end==1){
      up(mutexinspector);
      break;
    }
    up(mutexinspector);
  }
  up(guardia);
  sale_turista();
}

Turista(){
```

Ulises de los Llanos Valenzuela

19.)
```
Guardia_Parque() {
    noTuristas;
    down(mutexguardia);
    up(inicioAlarma);   flag=0;
    while(1){
        down(mutex inspector);
        if(flag==1){
            up(mutex inspector); down(mutex turistas); k = min(nTuristas, 20); if(k>0)
                nTuristas -= k; up(espera, k); up(mutex turista); flag=1;
            break;
        }
        up(mutex inspector);
        down(mutexTuristas);
        if(nTuristas >= 20){
            nTuristas -= 20;
            up(espera, 20);
            up(mutex Turistas);
            break;
        }
        up(mutex turistas);
    }
    up(mutexguardia);
    up(sale)
    if(flag != 1) goto noTuristas;
    sale_guardia();
}

Turista(){
    down(mutexturista);
    nTurista ++;
    up(mutexturista);
    down(espera);
    visitar();
}

Inspector(){
    while(1){
        down(inicioAlarma);
        alarm(15*60);
        down(sale); // sale el guardia
                    // o acaba la alarma
    }
}

sem mutexguardia = 1;
int flag = 0; // para si no hay turistas y acaba el tiempo
sem inicioAlarma = 0; // para inicializar el alarm
sem mutexInspector = 1;
sem mutexTurista = 1;
sem espera = 0; // barrera
sem sale = 0; // para despertar al inspector
```

20.)
```
cocheVa(){
    down(Va);
    down(mutexVa);
    nVa++;
    if(nVa == 1){
        down(paso);
    up(mutexVa);
    CruzarPuente();
    down(mutexVa);
    nVa--;
    if(nVa == 0)
        up(paso);
    up(mutexVa);
    up(Va);
}

cocheViene(){
    // es igual que coche Va
    // pero en vez de Va Viene
}

sem Va = 10;
sem mutexVa = 1;
int nVa = 0;
sem paso = 1;

main(){
    for(int i=1; i<20; i++){
        if( rand(coche()) ){ if(!fork()){cocheVa(); return 0;}
        else { if(!fork) cocheViene(); return 0;}
    }
}
```

Vicente de los Heros Valenzuela

21.) La solución es igual que el problema de los babuinos e igual que el ejercicio 20, pero inicializando nVa ~~a~~ y nVivre a 7.

Interbloqueo e Inanición

3.) se simula con ~~R~~ R32 t=1

$$R = \begin{pmatrix} 2 & 1 & 1 & 1 & 2 \\ 1 & 2 & 0 & 2 & 2 \\ 6 & 2 & 1 & 2 & 1 \\ \not1 & \not1 & \not0 & \not1 & \not1 \\ \not3 & \not0 & \not0 & \not0 & \not0 \end{pmatrix} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{matrix} \quad C = \begin{pmatrix} 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ \not0 & \not0 & \not0 & \not1 & \not1 \\ \not3 & \not0 & \not0 & \not0 & \not0 \end{pmatrix}$$

$A = (3\ 1\ 0\ 1\ 1)$

Se asigna P5 y acaba => $A = (6\ 1\ 0\ 1\ 1)$

Se asigna P4 y acaba => $A =$ ~~(7 2 0 2 2)~~

$(6\ 1\ 0\ 2\ 2)$

Y ya no se puede asignar más. También podríamos haber visto que al proceso 1 no se le puede dar otro recurso 3 porque solo hay 1 recurso 3 y lo tiene él, así que hay un bloqueo.

8.) A) $B = DM - C =$

$$\begin{pmatrix} \not6 & \not6 & \not0 & \not0 \\ \not0 & \not7 & \not5 & \not0 \\ 6 & 6 & 2 & 2 \\ \not7 & \not0 & \not0 & \not2 \\ \not0 & \not3 & \not2 & \not0 \end{pmatrix} \begin{matrix} P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{matrix}$$

B) $E_i = A_i + \sum_j C_{ji} \Rightarrow E = (6\ 7\ 12\ 12)$

Se libera P0 y acaba => ~~E~~ $A = (2\ 1\ 1\ 2)$

Se libera P3 y acaba => $A = (4\ 4\ 6\ 6)$

Se libera P4 y acaba => $A = (4\ 7\ 9\ 8)$

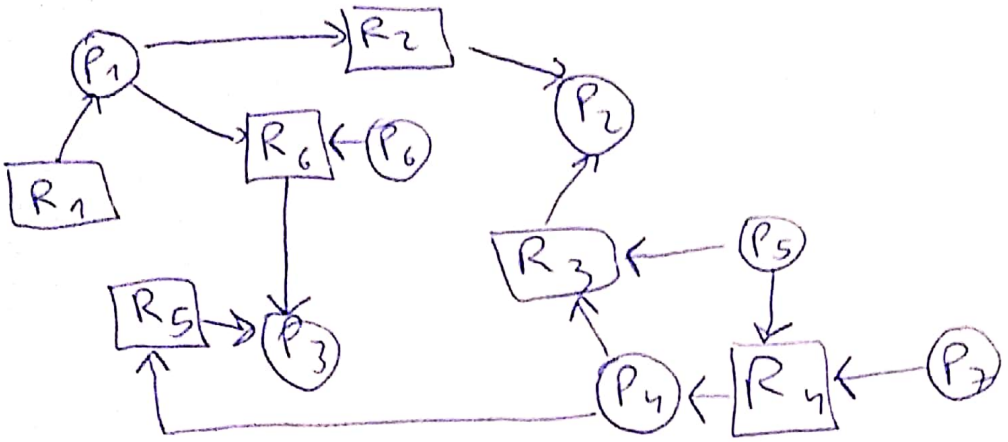
Se libera P1 y acaba => $A = (6\ 7\ 9\ 8)$

Scanned with CamScanner

C) Es un estado seguro porque todos los procesos han acabado.

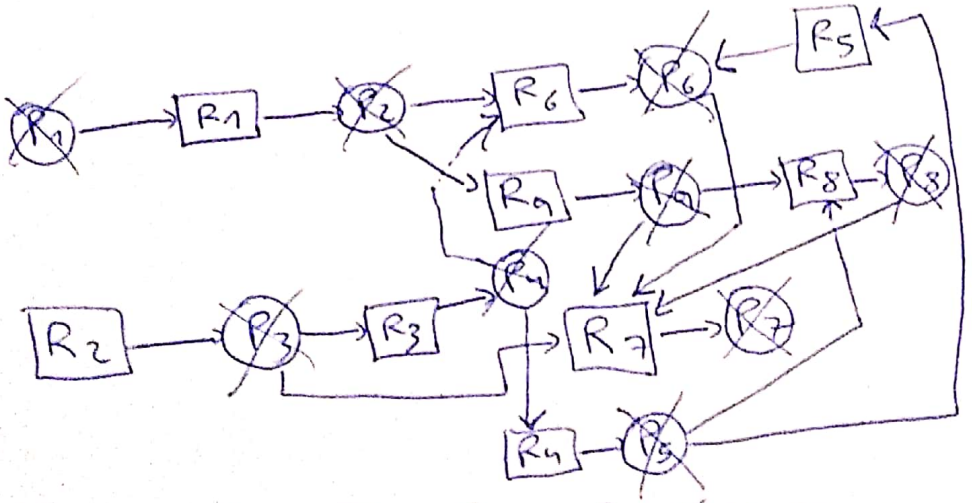
D) ~~No pueden quedarse bloqueados porque es un estado seguro~~
Si los recursos se reparten en el orden del algoritmo del banquero, no quedan procesos bloqueados, pero si se reparten en otro orden pueden quedar bloqueados.

11.) Dibuja el grafo de Recursos:



No hay ningún ciclo, así que el estado es seguro y no habrá interbloqueos. Todos los procesos pueden acabar.

12.) Dibuja el grafo de Recursos:



No hay ningún ciclo, así que el estado es seguro y no habrá interbloqueos. Todos los procesos pueden acabar.

Scanned with CamScanner

MEMÓRIA

12.)

A) 2^{48} B

B) 2^{56} B

C) 2^{36} páginas (48 - 12)

D) 52 - 12 = 40 ⇒ 2^{40} maneiras.

E) $\frac{2^{15}}{64} = 2^9$ endereços.

$64 \rightarrow 40 - 67$

F) 64 bits

G) $\left\lceil \frac{6 \cdot 2^{13} + 50 \cdot 2^{13}}{4 \cdot 2^{13}} \right\rceil = 1 5 4 9$

$\left\lceil \frac{1549}{512} \right\rceil = 4$ 3° nível

$\left\lceil \frac{4}{512} \right\rceil = 1$ 2° nível

$\left\lceil \frac{1}{512} \right\rceil = 1$ 1° nível

4 + 1 + 1 = 6 páginas

13.) $\left\lceil \frac{2^{13}}{2^{12}} \right\rceil + \left\lceil \frac{16386}{2^{12}} \right\rceil + \left\lceil \frac{15870}{2^{12}} \right\rceil =$

$= 8 + 5 + 4 = 17$

$\frac{2^{16}}{2^{12}} = 16$ páginas ⇒ Não cabe.

Scanned with CamScanner

$\frac{2^{16}}{512} = 128$ páginas.

$\left\lceil \frac{2^{15}}{512} \right\rceil + \left\lceil \frac{1638}{512} \right\rceil + \left\lceil \frac{15870}{512} \right\rceil =$

$64 + 33 + 31 = 128 \Rightarrow$ Sí que cabe.

14.] $\frac{128 \cdot 30 \cdot 2^{3}\,B}{256\,B} = 30$ fallos de página.

↙ 3 Bytes para el int

FICHEROS

4.] A.] $2^{16} \cdot 2^{5}\,KB = 2^{31}\,B$

B.] $\frac{2^{29}\,B}{2^{15}\,B} = 2^{14}$ bits $= 4\,2^{72}\,B \Rightarrow$ 1 B bloque

↑ pero

C.] $\frac{2^{20}\,B}{2^{15}\,kB} = 2^{5} \Rightarrow$ es entero, no hay fragmentación interna.

5.] A.] 2^{31} bloques

B.] $2^{31} \cdot 2\,kB = 2^{42}\,B$

C.] $\frac{2^{11}\,B}{64} = \frac{2^{11}}{2^{6}} = 2^{8}$ punteros en 1 bloque.

$(12 + 2^{8} + 2^{8}2^{8} + 2^{8}2^{8}2^{8})\,2^{11}\,B \simeq 2^{35}\,B$ el máximo

D.] Hay $\frac{2^{40}}{2^{13}} = 2^{27}$ i-nodos

Scanned with CamScanner

$2^{27} \cdot 128B = 2^{34}B$ ocupa la tabla (Tamaño de los ficheros Volumen)

E) Tantos como i-Nodos haya $\Rightarrow 2^{27}$ ficheros.

F) Tantos como bloques $\Rightarrow \frac{2^{40}B}{2^{11}B} = 2^{29}$ bits

8.)

A) $\frac{2^8 KB}{4B} = 2^{16}$ ptrs en 1 bloque.

$(8 + 2 \cdot 2^{16} + 2^{16} \cdot 2^{16} + 2^{16} 2^{16} 2^{16}) 256KB \simeq 2^{66}B$

B) $2^{64}B\ 2^{32}\ 2^{32}\ 2^{18} \cdot 2B = 2^{50}B$

C) No tiene sentido permitir ficheros más grandes que el máximo del disco

13.) El máximo permitido $\Rightarrow 2^{16} \cdot 2^{15}B = 2^{31}B = 2GB$

- De tamaño $\frac{2^{30}B}{2^{15}B} = 2^{15}$ bloques, ocupa 2^{19} bits $= 2^{16}B$

$126 \cdot 1024 \rightarrow$ *S2: 7 - 3 - 14 - 8

$828 \rightarrow$ *S3: 4

$80 \cdot 1024 \rightarrow$ *S4: 9 - 6 - 5

- Interna porque solo en el último bloque

- *S2: $4032 \cdot 2^{10} - 126 \cdot 1024 = 2048B = 2KB$

*S3: $32KB - 828KB = 31940KB$

*S4: $(3 \cdot 32 \cdot 2^{15} - 80 \cdot 1024)B = 16384B = 2^{14}B$.

Scanned with CamScanner