Pauta Auxiliar Recuperativo

Profesores: Luis Mateu

Auxiliares: José Astorga
Vicente González
Pablo Jaramillo

P1. Spinlocks

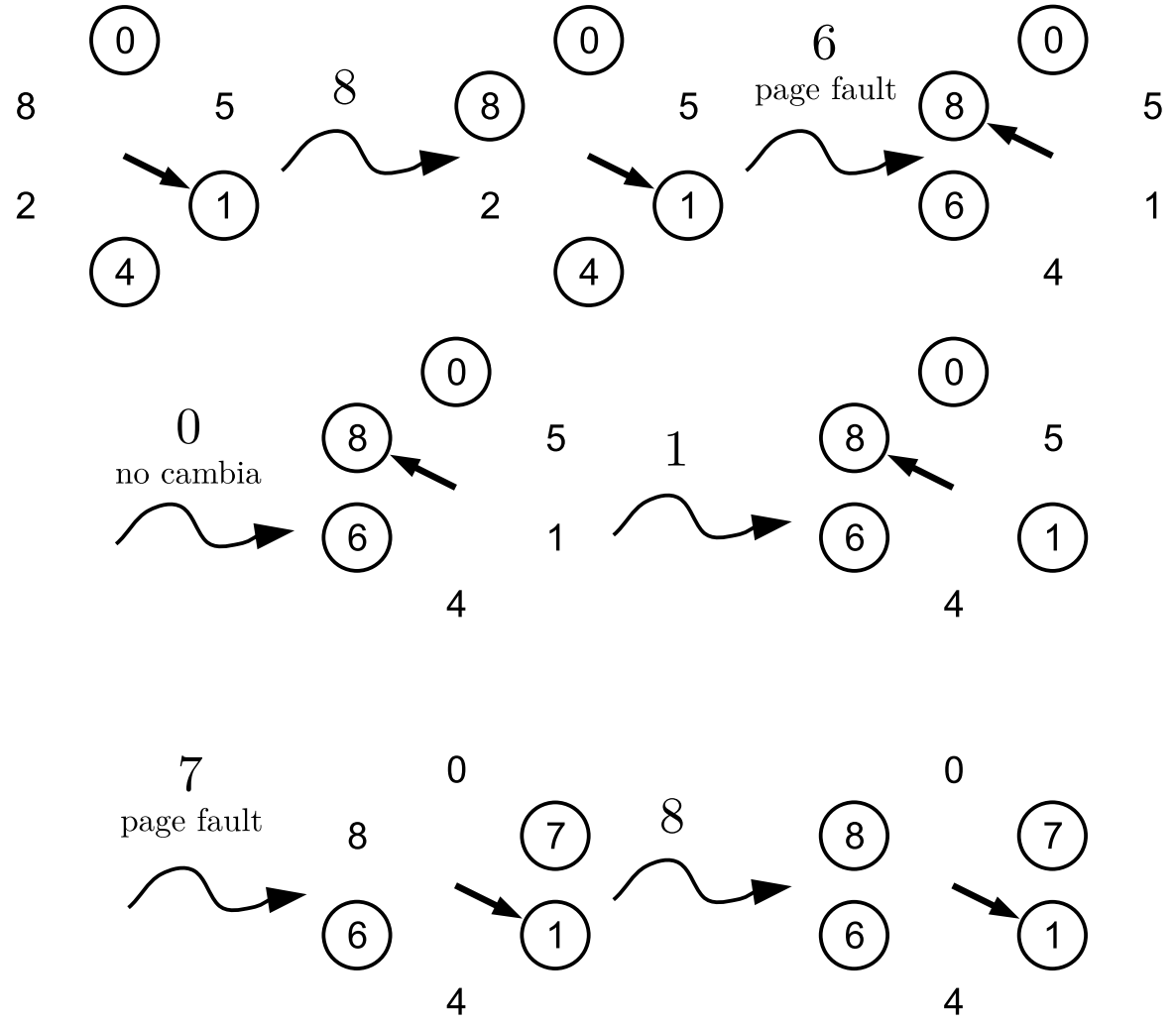
```
typedef struct {
  int id;
  int w;
} Req;

Queue *q;
int palitos[5];
int sl = OPEN;

void init() { /* ... */ }

void pedir(int id) {
  spinLock(&sl);
  if ( !palitos[id] && !palitos[(id+1)%5] &&
       emptyQueue(q) ) {
    palitos[id]= palitos[(id+1)%5]= 1;
    spinUnlock(&sl);
  }
  else {
    Req req= { id, CLOSED };
    put(q, &req);
    spinUnlock(&sl);
    spinLock(&req.w);
  }
}
void devolver(int id) {
  spinLock(&sl);
  palitos[id]= palitos[(id+1)%5]= 0;
  while (!emptyQueue(q)) {
    Req *preq= peek(q);
    int w_id= preq->id;
    // Condición para que thread continúe en espera:
    if (palitos[w_id] || palitos[(w_id+1)%5])
      break; // Si no puede comer, los que vienen después tampoco
    get(q);
    palitos[w_id]= palitos[(w_id+1)%5]= 1;
    spinUnlock(&preq->w);
  }
  spinUnlock(&sl);
}
```

P2. Estrategia Reloj



P3. Diagrama de Páginas

(a) Proceso A

Pág. virtual	Pág. real	V	W
0 → 3	-	0	-
4	2	1	0
5	7	1	0
6	-	0	0
7	5	1	1
8	**0**	**1**	**1**
9	**4**	**1**	**1**
10	**8**	**1**	**1**
11	10	1	1

(b) Proceso hijo de B

Pág. virtual	Pág. real	V	W
0 → 3	-	0	-
4	3	1	**0**
5	1	1	**0**
6	6	1	**0**
7,8,9,10	5	0	**-**
-			
-			
-			
11	**11**	1	**1**

Lo importante en la parte (b) es darse cuenta que al acceder a una página compartida, se asigna una copia de la original en una nueva página real disponible y que el bit W en el proceso que hizo la escritura quede en 1. Note que al asignar una página puede ser cualquiera de las que están disponibles

P4. MMU

```
Process current_process;

// Graba en disco la página page del proceso q
int pageOut(Process *q, int page) {
    int *qtab= q->pageTable;
    int realPage= getRealPage(qtab[page]);
    if (bitD(&qtab[page]))
        savePage(q, page); // retoma otro proceso
    setBitV(&qtab[page], 0);
    setBitS(&qtab[page], 1);
    return realPage; // Retorna la página real en donde se ubicaba
}
// Recupera de disco la página page del proceso q colocándola en realPage
void pageIn(Process *p, int page, int realPage) {
    int *ptab= p->pageTable;
    setRealPage(&ptab[page], realPage);
    setBitV(&ptab[page], 1);
    loadPage(p, page); // retoma otro proceso
    setBitD(&ptab[page], 0);
    setBitS(&ptab[page], 0);
    purgeTlb(); // invalida la TLB
    purgeL1(); // invalida cache L1
}
```

P5. Módulos

Si se fijan `copy_from_user` no avanza `inc_buf` y por lo tanto siempre copiaba el mismo byte:

shell 1	shell 2
$ echo > /dev/inc $ (termina)	
$ echo > /dev/inc (no termina)	
	$ cat /dev/inc $ (termina con 2 \n)
$ (termina)	$ cat /dev/inc aaaaaaaaaa (10 a, sin \n, no termina)
$ echo ef > /dev/inc $ (termina)	eeeeee (6 e, sin \n, no termina)
$ echo ghi > /dev/inc <control-C> (indefinido) $ (termina)	gggggggg (8 g, sin \n, no termina)

Sin embargo, si ignoramos este hecho, la solución válida sería:

shell 1	shell 2
$ echo > /dev/inc $ (termina)	
$ echo > /dev/inc (no termina)	
$ (termina)	$ cat /dev/inc aabbcc $ (termina con 2 \n)
	$ cat /dev/inc
$ echo ef > /dev/inc $ (termina)	eeff $ (termina con 2 \n)
$ echo ghi > /dev/inc <control-C> (indefinido) $ (no termina)	