Sistemas Operativos

Universidad Complutense de Madrid

2020-2021

Procesos y Planificación

Soluciones de problemas

Juan Carlos Sáez

Problema 4.a (I)

Problema de la implementación:

- Los procesos hijos heredan la visión lógica de los ficheros abiertos por el padre antes del `fork()`
  - Herencia de ficheros en UNIX
- Todos los procesos (padre e hijos) comparten el puntero de posición de ficheros origen y destino
  - Cada vez que un proceso mueve el puntero (vía `lseek()`,`read()` o `write()`) todos los demás ven la modificación del mismo

Problema 4.a (II)

Orden de ejecución (información del enunciado)

1. Padre ejecuta bucle completamente y se bloquea en la línea 23 (`wait()`)
2. Hijo 0 se ejecuta $\rightarrow$ copia un bloque
3. Hijo 1 se ejecuta $\rightarrow$ copia un bloque
4. Hijo 2 se ejecuta $\rightarrow$ copia un bloque
5. Hijo 3 se ejecuta $\rightarrow$ copia un bloque
6. Padre sale del bucle de la línea 23 y se ejecuta hasta el final

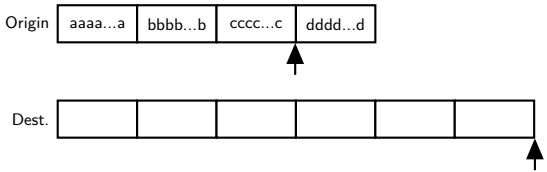
Problema 4.a (II)

Orden de ejecución (información del enunciado)

1. Padre ejecuta bucle completamente y se bloquea en la línea 23 (`wait()`)
2. Hijo 0 se ejecuta → copia un bloque
3. Hijo 1 se ejecuta → copia un bloque
4. Hijo 2 se ejecuta → copia un bloque
5. Hijo 3 se ejecuta → copia un bloque
6. Padre sale del bucle de la línea 23 y se ejecuta hasta el final

- Con este orden de ejecución la copia intercalada no se realiza correctamente
- Necesario analizar modificaciones del fichero destino paso a paso para poder obtener contenido del array `buf` del padre tras ejecutar sentencias de las líneas 24 y 26.

Problema 4.a (III)

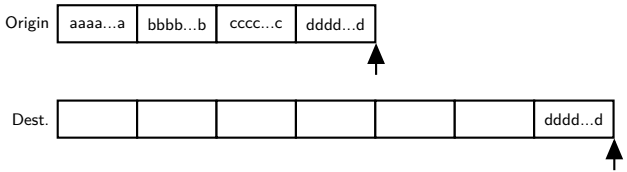



```
char buf[BLOCK]="xxxxxx...xxxx";

void copy_block(int fdo,int fdd) {
 read(fdo,buf,BLOCK);
 write(fdd,buf,BLOCK);
}
```

El padre deja al hijo 0 el puntero de posición del fichero origen al principio del bloque lógico 3 y el puntero de posición del fichero destino al principio del bloque lógico 6

Problema 4.a (III)

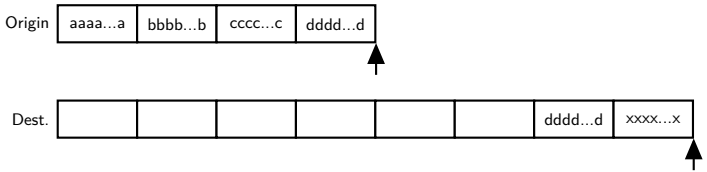



```
char buf[BLOCK]="xxxxxx...xxxx";

void copy_block(int fdo,int fdd) {
 read(fdo,buf,BLOCK);
 write(fdd,buf,BLOCK);
}
```

El hijo 0 transfiere el bloque lógico 3 del origen al destino. En teoría, debería haberse encargado de transferir el bloque 0.

Problema 4.a (III)

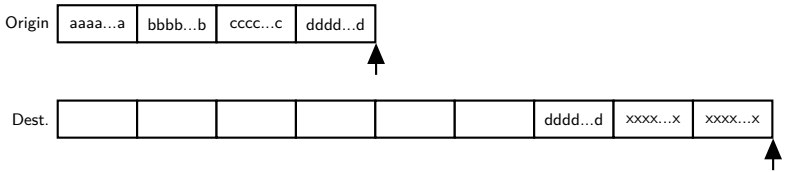



```
char buf[BLOCK]="xxxxxx...xxxx";

void copy_block(int fdo,int fdd) {
 read(fdo,buf,BLOCK);
 write(fdd,buf,BLOCK);
}
```

El hijo 1 intenta leer un bloque pero no lee nada (EOF). El array buf no se modifica al no leer nada y, por tanto, se escribe el contenido original de buf al final del fichero

Problema 4.a (III)

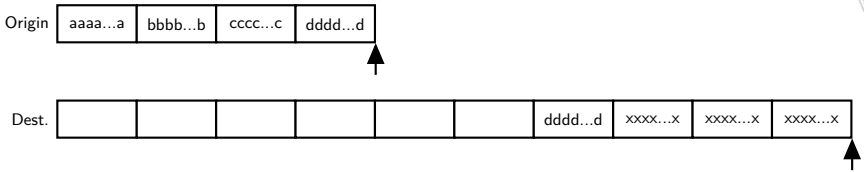



```
char buf[BLOCK]="xxxxxx...xxxx";

void copy_block(int fdo,int fdd) {
 read(fdo,buf,BLOCK);
 write(fdd,buf,BLOCK);
}
```

El hijo 2 tampoco lee nada del fichero origen (EOF) y, por tanto, escribe el contenido original de buf al final del fichero destino.

Problema 4.a (III)

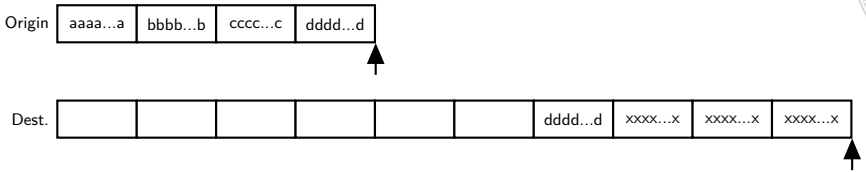



```
char buf[BLOCK]="xxxxxx...xxxx";

void copy_block(int fdo,int fdd) {
 read(fdo,buf,BLOCK);
 write(fdd,buf,BLOCK);
}
```

El hijo 3 realiza las mismas acciones que los hijos 1 y 2.

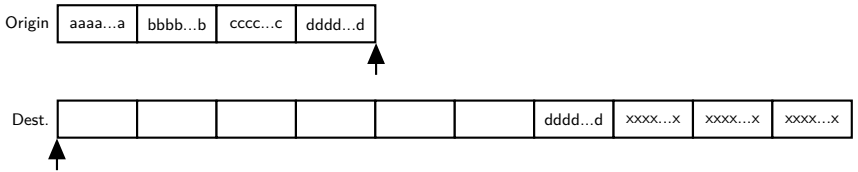
Problema 4.a (IV)




```
23: while (wait(NULL)!=-1) { };
24: read(fdd,buf,BLOCK);
25: lseek(fdd,0,SEEK_SET);
26: read(fdd, buf,BLOCK);
```

El padre ejecuta la sentencia de la línea 24 y lee 0 bytes (EOF). La variable buf no se modifica → `buf="xxxx..."`

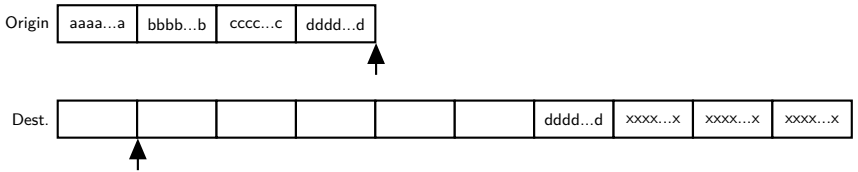
Problema 4.a (IV)




```
23: while (wait(NULL)!=-1) { };
24: read(fdd,buf,BLOCK);
25: lseek(fdd,0,SEEK_SET);
26: read(fdd, buf,BLOCK);
```

`lseek()` ubica el puntero de posición al principio del fichero.

Problema 4.a (IV)




```
23: while (wait(NULL)!=-1) { };
24: read(fdd,buf,BLOCK);
25: lseek(fdd,0,SEEK_SET);
26: read(fdd, buf,BLOCK);
```

El padre ejecuta la sentencia de la línea 26, leyendo un bloque completo del "hueco" dejado por `lseek()` (línea 16). Al leer del hueco, el SO devuelve ceros → `buf={0,0,0,0,0,…,0}`.

Problema 4.b

```
void copy_block(int i) {
 int fdo,fdd;

 fdo = open("Origin",O_RDONLY);
 fdd = open("Destination",O_WRONLY);
 lseek(fdo,i*BLOCK, SEEK_SET);
 lseek(fdd,2*i*BLOCK,SEEK_SET);
 read(fdo,buf,BLOCK);
 write(fdd,buf,BLOCK);
 close(fdo);
 close(fdd);
}
```

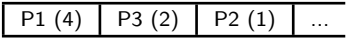
- Al reabrir el fichero desde cada proceso hijo, se obtiene una nueva representación lógica del fichero → punteros de posición privados

Problema 4.b (cont.)

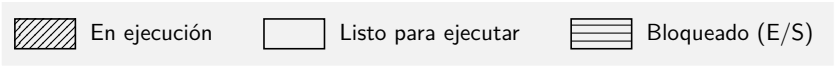
```
#define BLOCK 4096
char buf[BLOCK] = "xxxxxxx...xxxxx";
void main() {
 pid_t pid;
 int fdo,fdd,i;
 fdo = open("Origin",O_RDONLY);
 fdd = open("Destination",O_RDWR|O_CREAT|O_TRUNC,0666);
 for (i=0; i < 4; i++) {
   pid = fork();
   if (pid==0){
       copy_block(i);
       exit(0);
    }
 }
 while (wait(NULL)!=-1) { };
 read(fdd,buf,BLOCK);
 lseek(fdd,0,SEEK_SET);
 read(fdd, buf,BLOCK);
}
```

Problema 7: Consideraciones generales

- Dibujar el contenido de la *run queue* cuando el proceso actual abandona la CPU o es expropiado
  - Además, para planificadores expropiativos y SJF, incluir entre paréntesis la longitud de la ráfaga de CPU de cada proceso
  - Ejemplo:



- Por simplicidad, usar patrones para representar los estados de los procesos:
  - Ejemplo:



Problema 7: Consideraciones generales

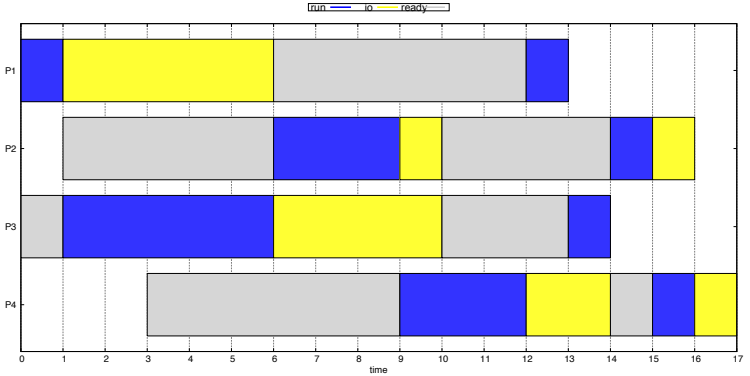
- Tened presente que puede haber varias soluciones posibles...
  - Cuando 2 o más procesos entran en el estado "Listo para ejecutar" al mismo tiempo
  - Debemos decidir el orden en el que los procesos se encolan (en caso de que no se proporcionen reglas o instrucciones específicas en el enunciado del problema)
    - Las decisiones tomadas (orden específico) se deben indicar explícitamente en la solución proporcionada

Recuerda: métricas

Métricas

- Tiempo de ejecución o de retorno $\Rightarrow$ $T_{ejecución} = T_{fin} - T_{creación}$
- Tiempo de espera: tiempo total que el proceso pasa esperando en la cola del planificador $\Rightarrow$ Tiempo total en estado "Listo"
- Productividad $= \dfrac{\#\ de\ procesos\ o\ tareas}{TiempoTotal}$
  - donde $TiempoTotal$ es el tiempo que tardan en completarse todas las tareas (desde el inicio hasta el final de la simulación)
- UsoCPU $= \dfrac{TiempoTotal - T_{idle}}{TiempoTotal} \cdot 100$
  - T_{idle} tiempo total que la CPU permanece inactiva (*idle*) durante toda la ejecución

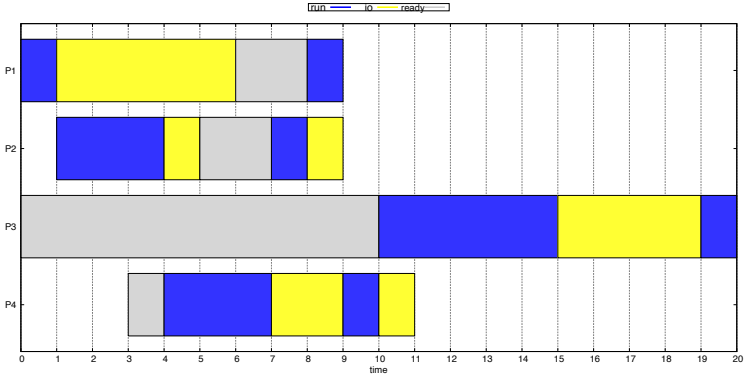
Problema 7 - FCFS



	P1	P2	P3	P4
T_{espera}	6	9	4	7
$T_{retorno}$	13	15	14	14

T_{espera_med}	=	6.5
$T_{retorno\_med}$	=	14
Productividad	=	0.235
UsoCPU	=	94.1 %

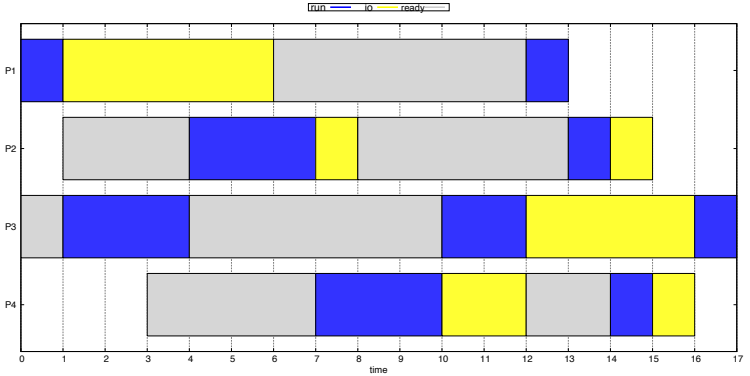
Problema 7 - SJF



	P1	P2	P3	P4
T_{espera}	2	2	10	1
$T_{retorno}$	9	8	20	8

T_{espera_med}	=	3.75
$T_{retorno\_med}$	=	11.25
Productividad	=	0.2
UsoCPU	=	80 %

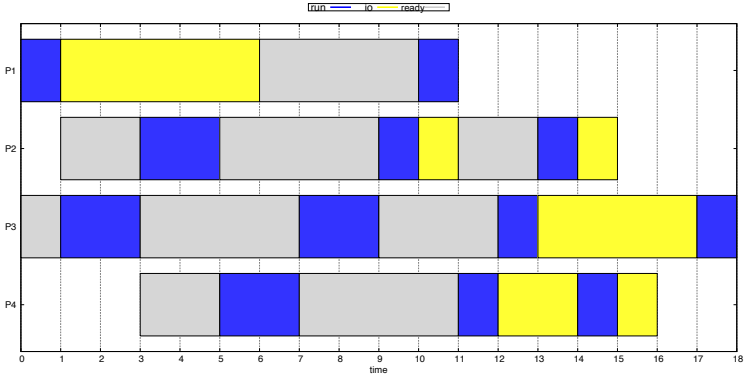
Problema 7 - RR (q=3)



	P1	P2	P3	P4
T_{espera}	6	8	7	6
$T_{retorno}$	13	14	17	13

T_{espera_med}	=	6.75
$T_{retorno\_med}$	=	14.25
Productividad	=	0.235
UsoCPU	=	94.1 %

Problema 7 - RR (q=2)



	P1	P2	P3	P4
T_{espera}	4	8	8	6
$T_{retorno}$	11	14	18	13

T_{espera_med}	=	6.5
$T_{retorno\_med}$	=	14
Productividad	=	0.22
UsoCPU	=	88.9 %

Problema 8

- 5 trabajos intensivos en CPU entran al sistema al mismo tiempo
  - *Cuanto mayor es el valor de prioridad, mayor es la prioridad*

Propiedad	**A**	**B**	**C**	**D**	**E**
T_{CPU}	10	6	2	4	8
Valor prioridad	3	5	2	1	4

- Calcular el tiempo de ejecución de cada proceso, así como el tiempos de ejecución medio:
  - **a.** RR con q $\approx$ 0
  - **b.** Por prioridad estricta
  - **c.** FCFS
  - **d.** SJF

Problema 8 - RR con q $\approx$ 0

Problema 8 - RR con q $\approx$ 0 (planificador teórico)

- Planificador completamente justo que garantiza un reparto equitativo de la CPU entre procesos con independencia del periodo de planificación considerado
  - Para cualquier intervalo de planificación T con n procesos listos para ejecutar y una sola CPU, cada proceso obtiene $\frac{T}{n}$ unidades de tiempo
- Para determinar el tiempo de ejecución de cada proceso, es preciso dividir la ejecución completa en "fases"
  - Una nueva "fase" comienza cuando uno de los procesos completa su ejecución

Problema 8 - RR con q $\approx$ 0

Problema 8 - RR con q $\approx$ 0 (planificador teórico)

- Como los procesos en este problema tienen un tiempo de CPU diferente, la ejecución se divide en 5 fases

5 activos	4 activos	3 activos	2 activos	1 activos
$T_{\#5}$	$T_{\#4}$	$T_{\#3}$	$T_{\#2}$	$T_{\#1}$

- Sea $T_{\#k}$ la cantidad de tiempo que tarda en ejecutarse el proceso más corto de los k procesos en ejecución
  - Como el planificador reparte de forma justa la CPU entre procesos activos, tenemos que:
    - $T_{\#k} = k \cdot min\left(T_{CPU}^{P1}, T_{CPU}^{P2}, \cdots, T_{CPU}^{Pk}\right)$
  - Cada proceso recibe el mismo tiempo de CPU durante el intervalo de planificación

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = ??$	$T_{\#4} = ??$	$T_{\#3} = ??$	$T_{\#2} = ??$	$T_{\#1} = ??$	$T_{retorno}$ (m.)
A						
B						
C						
D						
E						

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = 10$	$T_{\#4} = ??$	$T_{\#3} = ??$	$T_{\#2} = ??$	$T_{\#1} = ??$	$T_{retorno}$ (m.)
A	10					
B	6					
C	2					
D	4					
E	8					

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = 10$	$T_{\#4} = 8$	$T_{\#3} = ??$	$T_{\#2} = ??$	$T_{\#1} = ??$	$T_{retorno}$ (m.)
A	10	8				
B	6	4				
C	2	$--$				
D	4	2				
E	8	6				

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = 10$	$T_{\#4} = 8$	$T_{\#3} = 6$	$T_{\#2} = ??$	$T_{\#1} = ??$	$T_{retorno}$ (m.)
A	10	8	6			
B	6	4	2			
C	2	– –	– –			
D	4	2	– –			
E	8	6	4			

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = 10$	$T_{\#4} = 8$	$T_{\#3} = 6$	$T_{\#2} = 4$	$T_{\#1} = ??$	$T_{retorno}$ (m.)
A	10	8	6	4		
B	6	4	2	– –		
C	2	– –	– –	– –		
D	4	2	– –	– –		
E	8	6	4	2		

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = 10$	$T_{\#4} = 8$	$T_{\#3} = 6$	$T_{\#2} = 4$	$T_{\#1} = 2$	$T_{retorno}$ (m.)
A	10	8	6	4		
B	6	4	2	– –		
C	2	– –	– –	– –		
D	4	2	– –	– –		
E	8	6	4	2		

Problema 8 - RR con q $\approx$ 0

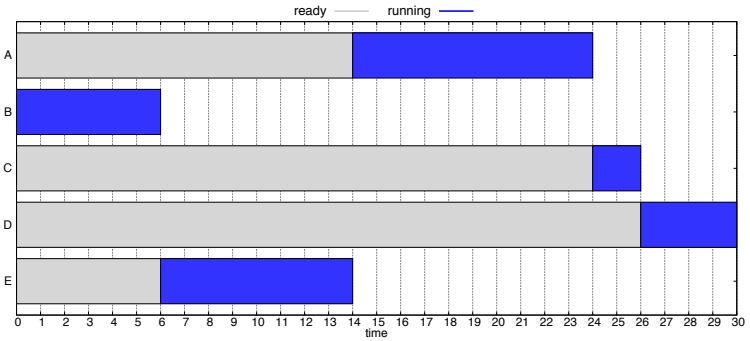
	$T_{\#5} = 10$	$T_{\#4} = 8$	$T_{\#3} = 6$	$T_{\#2} = 4$	$T_{\#1} = 2$	$T_{retorno}$ (m.)
A	10	8	6	4	2	
B	6	4	2	– –	– –	
C	2	– –	– –	– –	– –	
D	4	2	– –	– –	– –	
E	8	6	4	2	– –	

Problema 8 - RR con q $\approx$ 0

	$T_{\#5} = 10$	$T_{\#4} = 8$	$T_{\#3} = 6$	$T_{\#2} = 4$	$T_{\#1} = 2$	$T_{retorno}$ (m.)
A	10	8	6	4	2	$30m$
B	6	4	2	– –	– –	$24m$
C	2	– –	– –	– –	– –	$10m$
D	4	2	– –	– –	– –	$18m$
E	8	6	4	2	– –	$28m$

$T_{retorno\_med} = 22\text{m}$

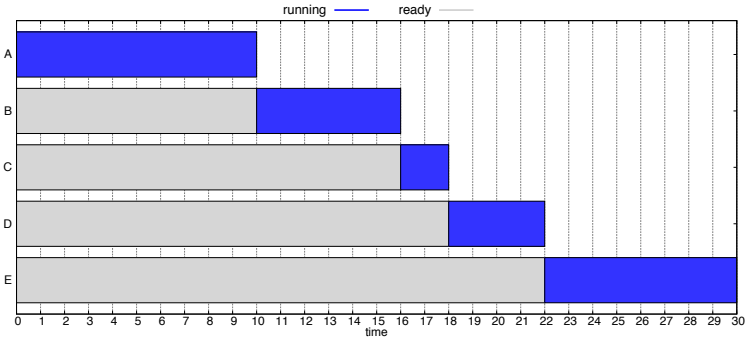
Problema 8 - Prioridad



Propiedad	**A**	**B**	**C**	**D**	**E**
$T_{retorno}$ (m.)	24	6	26	30	14
T_{CPU}	10	6	2	4	8
Valor prio.	3	5	2	1	4

$T_{retorno\_med} = 20m$

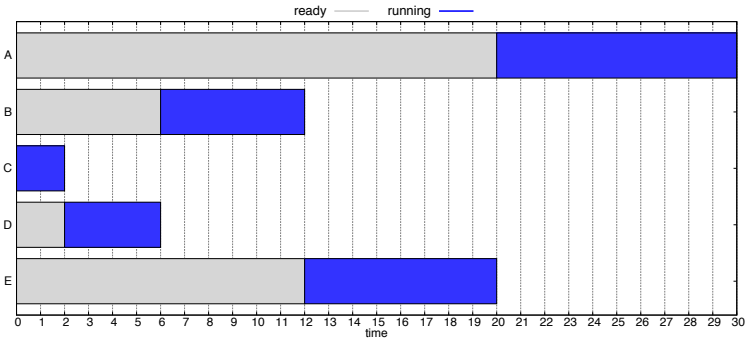
Problema 8 - FCFS



Propiedad	**A**	**B**	**C**	**D**	**E**
$T_{retorno}$ (m.)	10	16	18	22	30
T_{CPU}	10	6	2	4	8
Valor prio.	3	5	2	1	4

$T_{retorno\_med} = 19.2m$

Problema 8 - SJF



Propiedad	**A**	**B**	**C**	**D**	**E**
$T_{retorno}$ (m.)	30	12	2	6	20
T_{CPU}	10	6	2	4	8
Valor prio.	3	5	2	1	4

$T_{retorno\_med} = 14m$