Ingeniería Informática. Curso 3º.
Sistemas Operativos
Examen Final. PROBLEMAS.
2 de Febrero de 2010

1.a) Traza de ejecución temporal de los procesos (diagrama de Gantt):

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
P	0	1	2	2	2	2	2	3	3	1	1	1	1	1	4	4	4	0	0	0	4	0	0	0	

1.b) Indicar los tiempos medio de retorno y espera.

Proceso	Tiempo de retorno	Tiempo de espera
0	24	17
1	13	7
2	5	0
3	6	4
4	21	14
Tiempos medios	13,8	8,4

2. Se ofrecen dos soluciones distintas, la primera de ellas sin hacer uso de la función trywait(), y la segunda utilizando dicha función.

Solución sin trywait():
Inicialización:

aviso – semáforo a 0 */* semáforo binario*
pizzas – variable compartida a 0 */* necesita exclusión mutua*
mutex – semáforo a 1 */* implementa la región crítica de la variable pizzas*
pipe(fd);
close(fd[0]) en el Cocinero
close(fd[1]) en los Repartidores

Cocinero	*Repartidor*

`int buf[M];`
`while (TRUE) {`
`  COCINA PIZZAS y DESCANSA`
`  sem_wait(aviso);`
`  sem_wait(mutex);`
`  pizzas=M;`
`  sem_signal(mutex);`
`  write(fd, ptr, M*sizeof(int));`
`}`
	`Int buf;`
	`while (TRUE) {`
	`  sem_wait(mutex);`
	`  if (pizzas==0) sem_signal(aviso);`
	`  sem_signal(mutex);`
	`  read(fd, &buf, 1);`
	`  sem_wait(mutex);`
	`  pizzas--;`
	`  sem_signal(mutex);`
	`  SE VA A REPARTIR 1 pizza;`
	`}`

Solución usando trywait():
Inicialización:

pizzas – semáforo a 0 */* semáforo contador*
aviso – semáforo a 0 */* semáforo binario*
pipe(fd);
close(fd[0]) en el Cocinero
close(fd[1]) en los Repartidores

Cocinero	*Repartidor*

`int i=0;`
`char buf[M];`
`while (TRUE) {`
`  COCINA PIZZAS y DESCANSA`
`  sem_wait(aviso);`
`  write(fd, &buf, M);`
`  for (i=1; i<M; i++)`
`  sem_signal(pizzas);`
`}`
	`Char buf`
	`while (TRUE) {`
	`  if (sem_trywait(pizzas)==-1)`
	`  sem_signal(aviso);`
	`  read(fd, &buf, 1);`
	`  SE VA A REPARTIR 1 pizza;`
	`}`

Se realizan M-1 operaciones *signal*, ya que si el semáforo vale 0 no se decrementa.

3.a) Información relevante al estado de las tablas de descriptores, intermedia y de i-nodos:

Cada uno de los tres procesos que se crean tiene un Bloque Descriptor de Proceso (BCP) en el que se guarda SU tabla de ficheros abiertos, en la que el descriptor 0 es la entrada estándar, el 1 la salida estándar y el 2 la salida de errores; podemos suponer para el proceso P1 que el descriptor 3 (u otro superior) está libre en el momento de la llamada a la función "open". Ese descriptor será el que se asocie al fichero mediante la llamada "open", que creará un puntero a una tabla intermedia que contiene los atributos de apertura del fichero y el puntero de lectura/escritura, además de la referencia al i-nodo del fichero en memoria. De manera que se tendrá la siguiente información (los punteros son ejemplos):

Tabla de ficheros abiertos Proceso P1:

0	…
1	…
2	…
3	11
:	

Tabla de ficheros abiertos Proceso P2:

0	…
1	…
2	…
3	13
:	

Tabla de ficheros abiertos Proceso hijo de P2:

0	…
1	…
2	…
3	13
4	12

Tabla de punteros lec/esc intermedia

:	…	…	…
10	…		
11	R	0	70
12	W | A	50	77
13	RW	6	70
:	…	…	

Tabla de i-nodos de ficheros abiertos

:	…
70	I-nodo de "datos"
:	…
77	I-nodo de "textos"
:	

3.b) El contenido del fichero "datos" queda modificado, después de escribir en él los proceso P2 y su hijo (ver código); dado que estos dos procesos comparten el puntero de lectura/escritura se escribirá como sigue: *"abcabcA..A"*.

El contenido del fichero "textos" no se modifica.