Online

¿Cuando hay un cambio de contexto en SJF además de cuando se bloquea o termina?

B. En ninguno más.

Algoritmo optimiza CPU si no hay E/S

A. FIFO - Se usa menos el SO, esta en funcionamiento el proceso hasta que termina.

Cual falsa sobre EXEC

B. Si funciona bien devuelve 0 - Si funciona se ha cambiado la imagen.

Que señal permite enviar una señal a un proceso

C. kill

Que comparten A y B tras

```
if (fork()!=0)
  wait(&status);
else
  execve(B, parametros, 0);
```

A. descriptores de fichero.

Falso para pipes

C) Los dos procesos deben llamar a pipe.

* printf es una función, corresponde con write. Porello la llamada tiene nº var arg fijas.
* Padre e hijo comparten puntero en un fichero.

Redirige la salida a un fichero

A. close(1); creat(file, 0)

* La acción por defecto a una señal es muerte.
* ver la 14
* Un proceso zombie no consume CPU
* Un proceso solo puede cambiar de padre, cuando muere y lo hereda init.

Ejercicio 1. En un computador que utiliza política de planificación cíclica, se ejecutan los siguientes procesos de los que se conoce su tiempo de llegada y su tiempo total de ejecución:

Proceso	Tiempo de llegada	Duración
A	0	400
B	125	200
C	150	400
D	175	300

A) Rellene una tabla como la siguiente, asumiendo que la rodaja de tiempo es de 100 ms., indicando qué proceso está en ejecución y el estado de la cola de procesos listos en cada instante de tiempo. Indique en la columna eventos en qué momento del tiempo se produce para cada proceso, su llegada, su inicio y su finalización.

Instante	En ejecución	Cola de listos	Eventos
...	...	...	

B) Rellene una tabla como la siguiente, indicando para cada proceso los tiempos de llegada, servicio , inicio, fin, retorno, espera y espera normalizado.

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0						
B	125						
C	150						
D	175						

1) Rodaja 100 ms

a)

Instante	En ejecución	Cola de listos	Eventos
0	A(100)	—	Llegada A
100	A(200)	—	Continua A
200	A(300)	B, C, D	Entra B en 125, C 150 y D 175. Continua A
300	B(100)	C, D, A	Ejecuta B
400	C(100)	D, A, B	Ejecuta C
500	D(100)	A, B, C	Ejecuta D
600	A(400)	B, C, D	Fin A
700	B(200)	C, D	Fin B
800	C(200)	D	Continua C
900	D(200)	C	Continua D
1000	C(300)	D	Fin C
1100	D(300)	C	Continua D
1200	C(400)	—	Fin D

b)

Proceso	Llegada	Servicio	Inicio	Fin	Retorno	Espera	Retorno Normalizado
A	0	400	0	600	600	200	1'5
B	125	200	200	700	500	375	2'5
C	150	400	200	1000	800	650	2
D	175	300	200	1200	1000	825	3'33

Ejercicio 2. Dado el programa que se muestra a continuación:

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

main () {
  int pid,i;

  printf("INICIO\n");
  for (i=0; i<2; i++){
    pid=fork();
    if (pid == 0){
       sleep (1);
       printf("UNO\n");
       pid=fork();
       if (pid == 0) {
         printf("TRES\n");
         exit (0);
       }
    }
    else {
       sleep (2);
       printf(":DOS\n");
    }
  }
}
```

a) Indicar cuántas veces aparecen las palabras UNO, DOS y TRES en pantalla y en qué momentos de la ejecución, tomando como 0 el momento en el que se escribe INICIO.

b) Indicar la jerarquía de procesos creados utilizando la notación padre, hijo, nieto, bisnieto y mostrando claramente las relaciones jerárquicas y el orden de creación de los procesos dentro de su mismo nivel.

Ejercicio 3. Escriba una función en C sobre UNIX que permita ejecutar un mandato desde un programa. En caso de que no pueda ejecutarse el mandato, debe dar un mensaje de aviso y retornar al programa sin fallo. Posible declaración:

```
int sistema (char *nombre, char *argv)
```

Ejercicio 4. Codifique un programa en lenguaje C que genere dos procesos. El primer proceso lee números enteros del teclado y los envía a un pipe para que los lea el otro proceso. El segundo proceso recibirá los números del pipe sumándolos. Cuando el primer proceso lea el número 0 de teclado enviará una señal SIGALRM al proceso hijo para mostrar el resultado de la suma y terminar.

Ejercicio 5. Escriba en C un programa que cree la siguiente estructura de procesos:

El programa que se escriba será el que ejecute el proceso que se encuentra en el nivel 0. El programa deberá además cumplir los siguientes requisitos:

a) El proceso del nivel 0 esperará a que todos los procesos de los niveles 1 y 2 hayan terminado su ejecución.
b) Todos los procesos del nivel 1 y 2 deberán escribir por la salida estándar su identificador de proceso.

Ejercicio 6. Dos procesos A y B comparten el pipe `p`, siendo A el lector y B el escritor. En un momento determinado el pipe contiene 78 bytes y el proceso A está en ejecución, ejecutando las dos sentencias sucesivas siguientes sin que ejecute B:

```
read(p[0], buff1, 36);
read(p[0], buff2, 85);
```

¿Qué ocurre en las sentencias de lectura anterior?

3.)
```
int sistema (char *nombre, char *argv){
    int status = 0;
    pid = fork();
    switch (pid)
    {
      case 0:
        execvp(nombre, argv);
        perror("al hacer exec");
        exit(-1);
      case -1:
        status = -1;
        break;
      default:
        wait (&status);
    }
    return(status);
}
```

6.) pipe p A escritor
B lector 78bytes

78-36

read(p[0], buff1, 36); Quedaran 42 bytes

read(p[0], buff2, 85); Le pasa a B los 42 restantes

Lo de bloquearia si estubiera vacio, pero no es así hay 42B y A sigue siendo lector.

7.) Proceso 1 ⇒ Padre Proceso 2 y 3 ⇒ Hijos

1. Crear Tuberia
2. Crear hijo 2 y 3

```
int tuberia[2];
int pid1, pid2;
if(pipe(tuberia)<0){
    perror("No se puede crear la tubria");
    exit(0);
}
switch((pid1=fork())){
    case -1:
        perror("Error al crear el proceso")
        close(..
        close(..
        exit(0);
    case 0:
        close(tuberia[0]);
        break;
    default:
        switch(pid2=fork()){
            case -1:
                // Error el
            case 0:
                close(tuberia[1]);
                break;
```

Ejercicio 7. Escriba un programa que cree tres procesos que se conecten entre ellos utilizando una tubería tal y como se muestra en la siguiente figura:

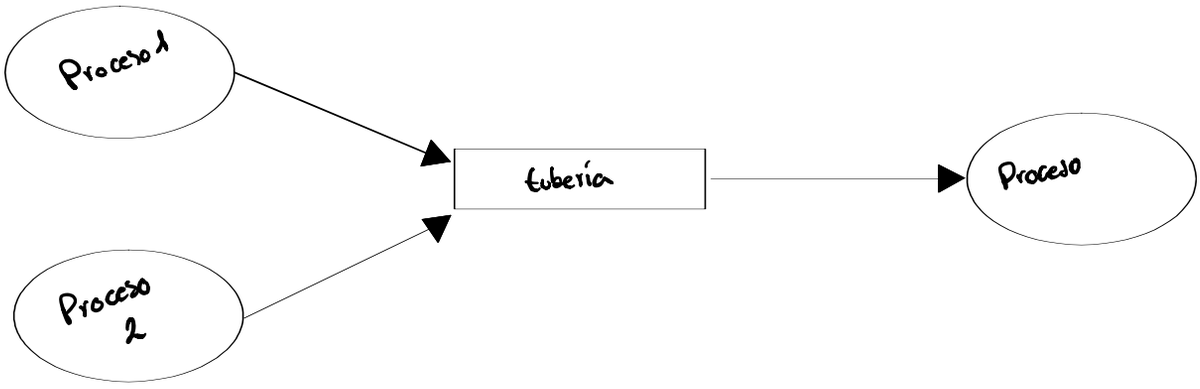
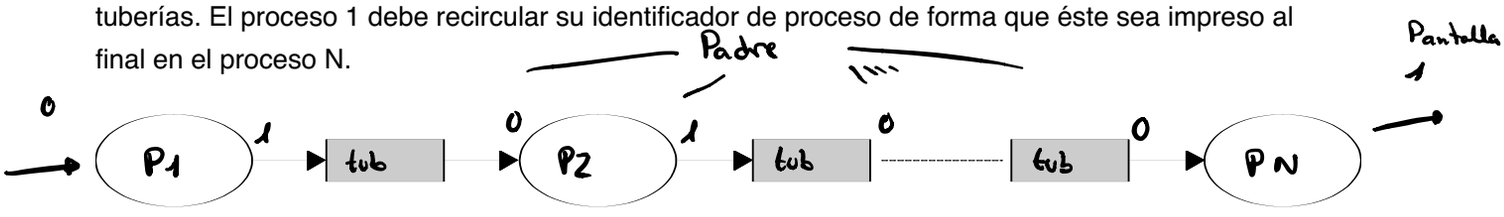


Figura 5.8 Procesos del ejercicio 5.15

Ejercicio 8. Escriba un programa que reciba un número entero por la línea de mandatos y cree la estructura de procesos conectados a través de tuberías que se muestra en siguiente figura, de forma que los procesos tengan su entrada y salida estándar redirigida a las correspondientes tuberías. El proceso 1 debe recircular su identificador de proceso de forma que éste sea impreso al final en el proceso N.



Ejercicio 9. Partiendo de los procesos creados en el ejercicio anterior, modifique el programa de forma que el proceso 1 genere 1000 números pares y el proceso 2 otros 1000 números impares. Tanto el proceso 1 como el proceso 2 introducen estos números en la tubería de forma que el proceso 3 los extrae y lo imprime por pantalla. El programa desarrollado debe asegurar que en la tubería nunca se insertan dos números pares seguidos o dos números impares seguidos.

8.)