SISTEMAS OPERATIVOS - CUESTIONES
13 de Febrero de 2015

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Cuestión 1. (0,75 puntos) Se dispone de un sistema de ficheros tipo UNIX (basado en nodos-i). En el directorio `/var/data` hay un fichero llamado `mensaje` que contiene el texto "*Tres más dos no es igual que cuatro más uno*". Estando en el directorio `/var`, el usuario crea un nuevo directorio `/var/backup` y, dentro del nuevo directorio, crea un enlace rígido (físico) al fichero `mensaje` llamado `old`. Posteriormente, desde ese mismo directorio `backup`, crea un enlace simbólico al fichero `mensaje` llamado `symOld`. Finalmente, el usuario borra el fichero `mensaje` del directorio `/var/data`. Conteste a las siguientes preguntas:

a. Usando **cuatro** comandos *shell*, realice las acciones descritas (asumir que el directorio actual es `/var`)

b. ¿Cuántos nodos-i nuevos se han utilizado tras ejecutar esos cuatro comandos? Justifique su respuesta

c. ¿Qué se mostrará por pantalla al ejecutar los siguientes comandos:

```
> cat /var/backup/old


> cat /var/backup/symOld
```

Cuestión 2. (0,75 puntos) Un determinado módulo Linux se inicializa del siguiente modo:

```
#define MAJOR 249
#define DEVICE_NAME "printer"
static int major;
static struct
file_operations fops = {
  .read = device_read,
  .write = device_write,
  .open = device_open,
  .release = device_release
};
```

```
int init_module(void) {
  major = register_chrdev(MAJOR, DEVICE_NAME, &fops);
  if (major != MAJOR ) {
    printk(KERN_ALERT "Reg. failed for %d\n", MAJOR);
    return major;
  }
  // ...
}
```

Para trabajar con el módulo se crea un fichero de dispositivo con el siguiente comando: `mknod /dev/printer c 249 0`

(a) Explica el significado de cada uno de los parámetros de este comando.

(b) Indica cuál o cuáles de las funciones registradas por register_chrdev se ejecutarán cuando el usuario ejecute el comando `echo "My printer is working!" > /dev/printer`

Cuestión 3. (0,75 puntos) Se desea imprimir una cadena en una impresora. Para ello, se ha creado un fichero de dispositivo y un módulo. Para imprimir en la impresora, primero se consulta el estado de la misma usando la dirección `PRINTER_STATUS_REG`. Cuando el contenido de la dirección es igual a la constante `READY`, se escribe un carácter en la dirección `PRINTER_DATA_REG`. Esto se repite para cada carácter. Suponiendo que las constantes `PRINTER_STATUS_REG`, `PRINTER_DATA_REG` y `READY` ya están definidas globalmente, completar la siguiente función para que al escribir una cadena en el fichero de dispositivo, la impresora imprima dicha cadena correctamente:

```
static ssize_t device_write(struct file *filp, const char *buffer,
                            size_t len, loff_t *offp) {
  ... */




  /* ... completando además "los huecos" del siguiente código */
  for (____________________) {
    while (inb(PRINTER_STATUS_REG) != READY) { }
    ____________________________;
  }
  return len;
}
```

Cuestión 4. (0,75 puntos) En nuestro sistema operativo disponemos de cerrojos y variables condicionales, pero no de semáforos. Respetando la semántica de `sem_wait(sem_t* s)` y `sem_post(sem_t* s)` vistas en clase, propón una implementación de esas dos funciones usando cerrojos y variables condicionales (completa el tipo `sem_t` como consideres oportuno). La implementación propuesta puede sufrir inanición (se valorará igualmente).

```
typedef struct _sem_t
{
   lock_t mutex;
   cond_t vc;




}sem_t;
```

```
int sem_wait(sem_t* s) {




}
```

```
int sem_post(sem_t* s) {




}
```

Cuestión 5. (0,75 puntos) En un sistema de paginación por demanda se obtiene que, con cierta carga de trabajo, la CPU se emplea un 15% del tiempo y el disco de swap está ocupado el 92% del tiempo. ¿Cuál de estas acciones aumentaría más la utilización de la CPU? a) Ampliar la memoria principal; b) Aumentar el grado de multiprogramación; c) Cambiar el disco de swap por otro de más capacidad; d) Cambiar la CPU por otra más rápida. Razone la respuesta.

Cuestión 6. (0,75 puntos) Considerar la siguiente secuencia de referencias a direcciones de memoria virtual generadas por un sólo programa en un sistema con paginación pura:

`0x10, 0x31A, 0xF4, 0x17C, 0x47C, 0x3B8, 0x284, 0x4FE, 0x4C, 0x334, 0x158, 0x26D, 0x502`

a. Deducir la cadena de referencias (secuencia de números de página generadas por el programa), suponiendo un tamaño de página de 256 Bytes.

b. Determinar razonadamente el número de fallos de página usando como estrategia de sustitución el algoritmo del reloj, suponiendo que hay cuatro marcos de página disponibles para el programa y que están inicialmente vacíos.

REF.	0x10	0x31A	0xF4	0x17C	0x47C	0x3B8	0x284	0x4FE	0x4C	0x334	0x158	0x26D	0x502
#pag													
m0													
m1													
m2													
m3													

Cuestión 7. (0,75 puntos) Considere el siguiente código:

```
#define N 8
int A[N] = {0};

void* suma(void* arg) {
 int* params = (int*) arg;
 int i, sum=0;
 for (i=params[0]; i <= params[1]; i++)
        sum += A[i];

 params[0] = sum;
 return NULL;
}
```

```
int main(int argc, char* argv[])
{
 int pid,n,i,fd;
 pthread_t th1, th2;
 int arg1[2], arg2[2];
 pid = fork();
 if (pid == 0) {
      for (i=0; i<N; i++)
           A[i] = i;
 }
 else {
     wait(NULL);
 }
 arg1[0] = 0;   arg1[1] = N/2 - 1;
 arg2[0] = N/2; arg2[1] = N-1;
 pthread_create(&th1, NULL, suma,(void*) arg1);
 pthread_create(&th2, NULL, suma,(void*) arg2);
 pthread_join(th1, NULL);
 pthread_join(th2, NULL);
printf("pid=%d ppid=%d resultado: %d + %d = %d\n",
getpid(), getppid(), arg1[0], arg2[0], arg1[0] +
arg2[0]);
 return 0;
}
```

Asumiendo que el proceso original tiene PID 100 (y es hijo de *Init*) y que al hijo se le asigna el PID 101, explique razonadamente cuántos hilos se llegan a ejecutar concurrentemente y qué mensajes se mostrarán por pantalla.

Cuestión 8. (0,75 puntos) Considere el siguiente código:

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/wait.h>
char buf[11] = "XXXXXXXXXX";
int main() {
        int fd1,fd2;
        fd1=open("prueba",O_WRONLY | O_CREAT | O_TRUNC, 0666 );
        fd2=open("prueba",O_RDONLY);
        if ( fork() == 0) {
                close(fd2);
                write(fd1,"AAAAA",5);
                close(fd1);
        }
        else {
                close(fd1);
                wait(NULL);
                read(fd2,buf,11);
                buf[10] = '\0';
                execlp("/bin/echo", "/bin/echo", buf, NULL);
        }

        printf("%s\n",buf);
        return 0;
}
```

Indique si hay algún error en el programa anterior o si por el contrario es correcto. Indique qué se verá en la consola si lo ejecutamos. Si considera que hay más de una posibilidad, discutirlas todas. Indique qué proceso imprime cada uno de los mensajes. **Justifique la respuesta**.

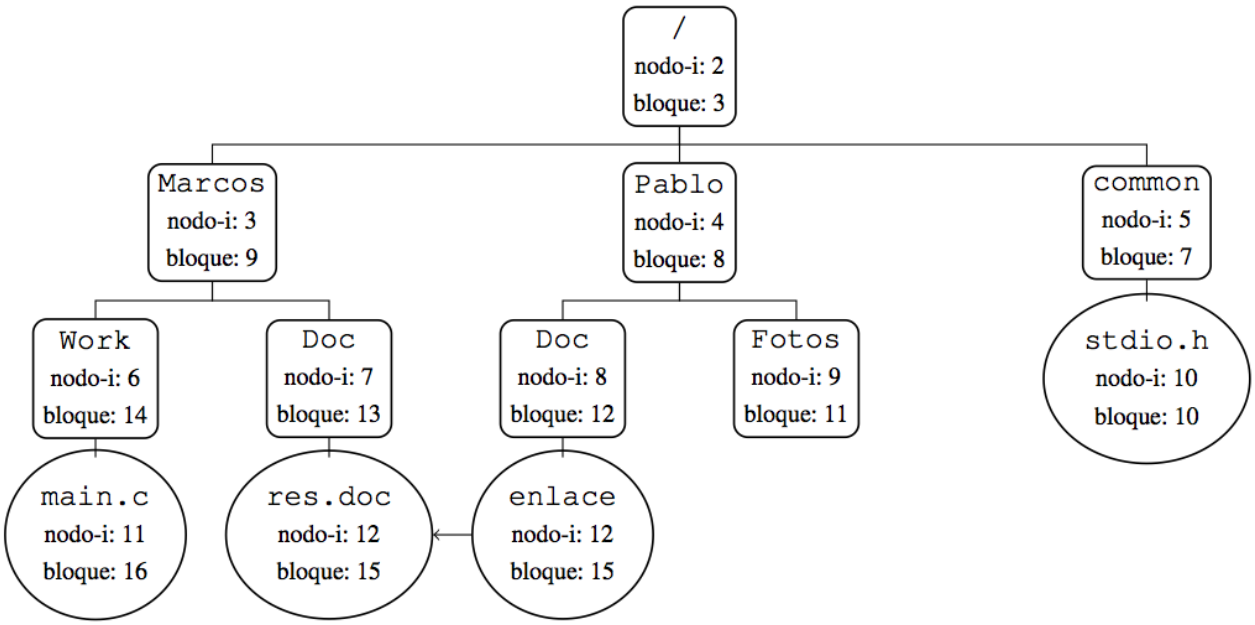
SISTEMAS OPERATIVOS - PROBLEMAS
13 de Febrero de 2015

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Problema 1. (2 puntos) Un sistema de ficheros tipo UNIX utiliza bloques de disco de 2K bytes. Para el direccionamiento de estos bloques se utilizan punteros de 32 bits. Para indicar el tamaño del fichero y el desplazamiento (offset) de la posición en bytes en las operaciones read() y write(), se utilizan números de 64 bits. Cada nodo-i tiene 10 punteros de direccionamiento directo, 1 puntero indirecto simple y 1 puntero indirecto doble.

a. ¿Cuál será el tamaño máximo de un fichero en este sistema suponiendo despreciable el espacio ocupado por el superbloque y la tabla de nodos-i?
b. ¿Cuál sería el porcentaje de espacio desaprovechado si el volumen donde está instalado el sistema de ficheros estuviese completamente lleno de ficheros de 7KB cada uno?
c. Dada la siguiente estructura de directorios[1], en el que Pablo comparte el fichero *res.doc* de Marcos mediante un enlace rígido de nombre *enlace*, indicar los contenidos de los directorios y de los nodos-i que encontrará el sistema (y el orden en que se los encontrará) al hacer la búsqueda del fichero *enlace* desde el directorio raíz.



[1] Los directorios se muestran como rectángulos con bordes redondeados y los ficheros regulares como óvalos o círculos.

Problema 2. (2 puntos) En un buffet se sirven pizzas de dos tipos: *pepperoni* (0) y *margarita* (1). Cuando un cliente desea comer una pizza de un determinado tipo ha de ir a recogerla al buffet. Si en ese instante quedan pizzas del tipo elegido, el cliente se servirá una (invocando `retirarPizzaBuffet(tipoPizza)`) y comerá. En otro caso, el cliente deberá avisar al camarero del buffet, que estará bloqueado en ese momento. Al ser despertado, el camarero mirará qué tipo de pizzas hay que reponer, repondrá N unidades del tipo o tipos que falten (invocando `reponerPizzas(N,tipoPizza)`) y volverá a bloquearse. Entonces el cliente podrá servirse la pizza y comer.

Un número arbitrario de clientes y el camarero (hilos del programa concurrente) se comportan del siguiente modo:

```
void Cliente(int tipoPizza){
    while (true) {
       conseguirPizza(tipoPizza);
       comer();
    }
}
```

```
void Camarero(){
    while (true) {
       servirPizzas();
    }
}
```

Implementar las funciones `conseguirPizza()` y `servirPizzas()` empleando un mutex, variables condicionales y variables compartidas.

Se ha de tener en cuenta que sólo es seguro invocar `retirarPizzaBuffet(tipoPizza)` si quedan pizzas del tipo correspondiente. Análogamente, sólo se ha de invocar `reponerPizzas(N,tipoPizza)` si no quedan pizzas de ese tipo en el buffet.