	EXAMEN FINAL SISTEMAS OPERATIVOS 13 de febrero de 2015
	Apellidos, Nombre: _____ NIF: _____.

Cuestión 1 (1pt) En nuestro sistema operativo disponemos de cerrojos y variables condicionales, pero no de semáforos. Proponga una implementación las siguientes funciones usando cerrojos y variables condicionales:

```
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 2 (1pt) 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. Para aumentar la utilización de la CPU se estudian las siguientes acciones:

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.

¿Cuál de ellas aumentaría más la utilización de la CPU? **Razone la respuesta**

Cuestión 3 (1pt) Considere 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) Deduzca la cadena de referencias (secuencia de números de página generadas por el programa), suponiendo un tamaño de página de 256 Bytes.

b) Determine 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.

Cuestión 6 (1pt) Suponga que un manejador de disco tiene la siguiente cola de peticiones de pistas pendiente:

87 – 24 – 55 – 2 – 9 – 35 – 72 – 23

Supóngase además que la última tanda de peticiones atendidas por el controlador fue 42 y 44 y que el disco tiene 120 pistas. Indique el orden de servicio y el desplazamiento de la cabeza lectora siguiendo las políticas: **FIFO**, **SSF**, **SCAN** y **C-SCAN**.

Cuestión 4 (1pt) 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;
}

void main(){
  int n, i, fd, arg1[2], arg2[2];
  pthread_t th1, th2;
```

```
  if (fork() == 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]);
}
```

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 5 (1pt) Considere el siguiente código:

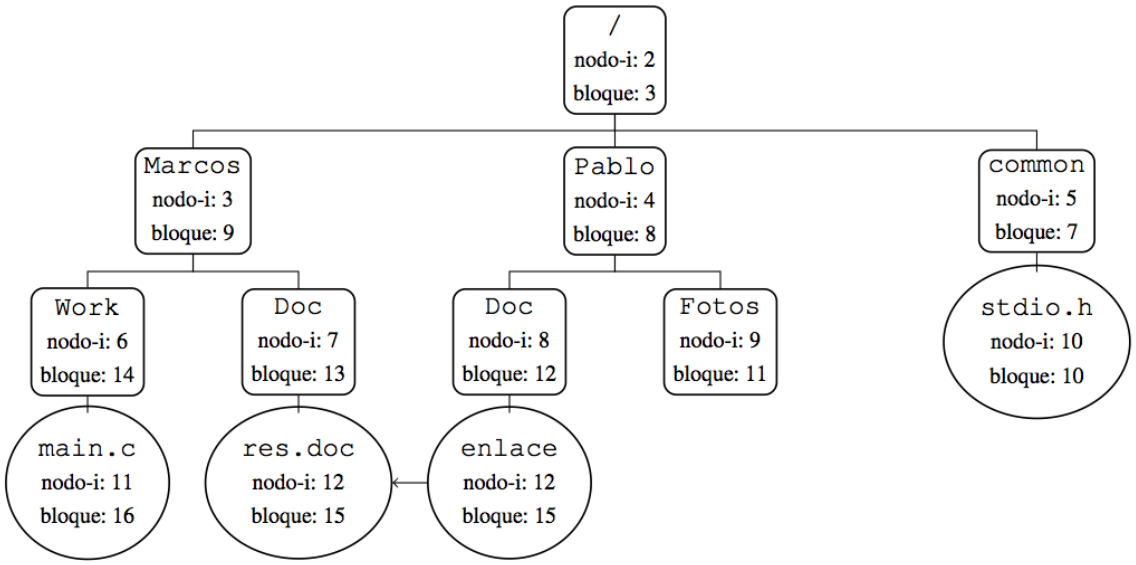
```
char buf[11] = "XXXXXXXXXX";
void main(void) {
  int fd1,fd2;
  fd1=open("prueba",O_WRONLY | O_CREAT | O_TRUNC, 0666 );
  fd2=open("prueba",O_RDONLY);
  if ( fork() == 0) {
    close(fd2);
    write(fd1,"AAAAA",5);
    close(fd1);
  } else {
    close(fd1);
    wait(NULL);
    read(fd2,buf,11);
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

  printf("pid: %d, buf: %s\n",getpid(), buf);
}
```

Indique si hay algún error (terminación abrupta del proceso) en el programa anterior o si por el contrario es correcto. Indique qué se verá en la consola si lo ejecutamos asumiendo que el proceso original tiene PID 100 y que al hijo se le asigna el PID 101. Si considera que hay más de una posibilidad, discuta brevemente todas las alternativas. **Justifique la respuesta**.

Problema 1. (2 pts) Un sistema de ficheros tipo UNIX utiliza bloques de disco de 2KiB. 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 7KiB cada uno?
c) Dada la siguiente estructura de directorios, en el que Pablo comparte el fichero `res.doc` de Marcos mediante un enlace rígido de nombre `enlace`, indique 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.



NOTA: Los directorios se muestran como rectángulos con bordes redondeados y los ficheros regulares como óvalos o círculos

Problema 2. (2 pts) En un buffet se sirven pizzas de dos tipos: `pepperoni` y `margarita`. 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();
  }
}
```

Implemente 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.