Examen de Sistemas Operativos.
20 de septiembre de 2007.

NOTAS:

* La fecha de publicación de las notas, así como de revisión se notificarán por Aula Global
* Para la realización del presente examen se dispondrá de **3 horas**
*** No** se pueden utilizar libros **ni** apuntes, ni usar móvil (o similar)
* Será necesario presentar el DNI o carnet universitario para realizar la entrega del examen

---

Ejercicio 1 (1,5 puntos) Conteste a las siguientes cuestiones:

1.a ¿Dónde reside el estado de un proceso cuando éste se está ejecutando?

En los registros del procesador.

1.b En el contexto de la planificación de procesos ¿Cuál es la función del planificador?

Seleccionar el siguiente proceso a ejecutar.

1.c ¿Qué es un proceso *zombie* en POSIX?

Es un proceso que finaliza (llamada *exit*) y para el que su proceso padre no ha realizado una llamada *wait*.

Ejercicio 2 (2 puntos) Indique las **llamadas al sistema** que se realizan para ejecutar el siguiente mandato:

ls –la |wc –l > fichero

NOTA: Sólo deben enumerarse las llamadas a las funciones del sistema (fork, open close,...), y sus parámetros, según el siguiente formato:

P0	P1	P2	….	Pn

Solución:

P0	P1	P2	….	Pn
newout=creat() pipe(pd) pid1=fork() pid2=fork() close(pd[0]) close(pd[1]) wait(pid2)	 close(pd[0]) close(newout) close(1) dup(pd[1]) close(pd[1]) execvp(”ls -la...”)	 close(pd[1]) close(0) dup(pd[0]) close(pd[0]) close(1) dup(newout) close(newout) execvp(”wc-l...”)		

Ejercicio 3 (3,5 puntos) Se desea escribir una aplicación que recibe peticiones a través de una tubería. Cada petición tiene un tamaño fijo de 255 caracteres que contendrá el nombre de un archivo.
Cada vez que recibe una petición, la aplicación coloca los datos de dicha petición en un búfer acotado. La aplicación utiliza cuatro hilos para procesar las peticiones. Cada vez que un hilo está libre, toma una petición del búfer acotado y realiza el procesamiento de la petición. El procesamiento de la petición consiste en determinar la longitud en bytes del archivo y poner dicha longitud en un búfer acotado de respuestas. Posteriormente habrá otro hilo que se encarga de enviar por un pipe los valores del búfer de respuesta.

Se pide:

a) Escriba las declaraciones de las estructuras de datos necesarias para la sincronización de los hilos, asumiendo que se ha decidido utilizar semáforos. **[0,5 puntos]**
b) ¿Qué otras variables globales serán compartidas por los hilos? **[0,5 puntos]**
c) Escriba el código del hilo *recibePeticiones*, suponiendo que dispone de una función obtenPeticion(char * nombreArch), que deja en la cadena nombreArch el nombre del archivo a leer. Si la petición recibida es la cadena vacía, el hilo debe terminar. **[1 punto]**
d) Escriba el código del hilo *procesaPeticion*, que obtiene peticiones del búfer acotado de peticiones y deja los resultados en el búfer acotado de respuestas. **[1,5 puntos]**

a)

```
sem_t    huecosPeticiones,    elementosPeticiones,    huecosRespuestas,
elementosRespuestas;
```

a)

```
#define MAX_BUFFER_PETICIONES 100
#define MAX_BUFFER_RESPUESTAS 200
char bufferPeticiones[MAX_BUFFER_PETICIONES][256];
char bufferRespuestas[MAX_BUFFER_RESPUESTAS[[256]
```

c)

```
void recibePeticiones(void) {
      int pos = 0;  /* posicion dentro del buffer */
      int i;
      char peticion[256];
      while (1) {
            obtenPeticion(peticion);
            if (strlen(petición)==0) break;
            sem_wait(&huecosPeticiones);  /* un hueco menos */
            strcpy(bufferPeticiones[pos], peticion);
            pos = (pos + 1) % MAX_BUFFER_PETICIONES;
            sem_post(&elementosPeticiones);/* un elemento mas */
      }
      pthread_exit(0);
}
```

d)

```
void procesaPeticion(void) {
      int posPet = 0;
      int posResp = 0;
      char petición[256];
      struct stat statbuf;
      off_t longitud;
      while (1) {
            sem_wait(&elementosPeticiones);    /* un elemento menos */
            strcpy(petición, buffer[pos]);
            pos = (pos + 1) % MAX_BUFFER_PETICIONES;
            sem_post(&huecosPeticiones);    /* un hueco mas */

            if (stat(petición, &statbuf) != 0) {
                  longitud=0;
            }
            else {
                  longitud = statbuf.st_size;
            }

            sem_wait(&huecosRespuestas); /* un hueco menos */
            bufferRespuestas[posResp] = longitud;
            pos = (pos + 1) % MAX_BUFFER_RESPUESTAS;
            sem_post(&elementosRespuestas);
      }
}
```

Ejercicio 4 (3 puntos) Dado el siguiente conjunto de ficheros y directorios en UNIX

Permisos	**Nombre fichero**
drwxr_xr_x	/
drwxr-xr-x	/bin
_rwxr_xr_x	/bin/ls
drwxr-xr-x	/usr
drwxr-xr-x	/home
drwxr_xr_x	/home/dir1
drwxr_xr_x	/home/dir2

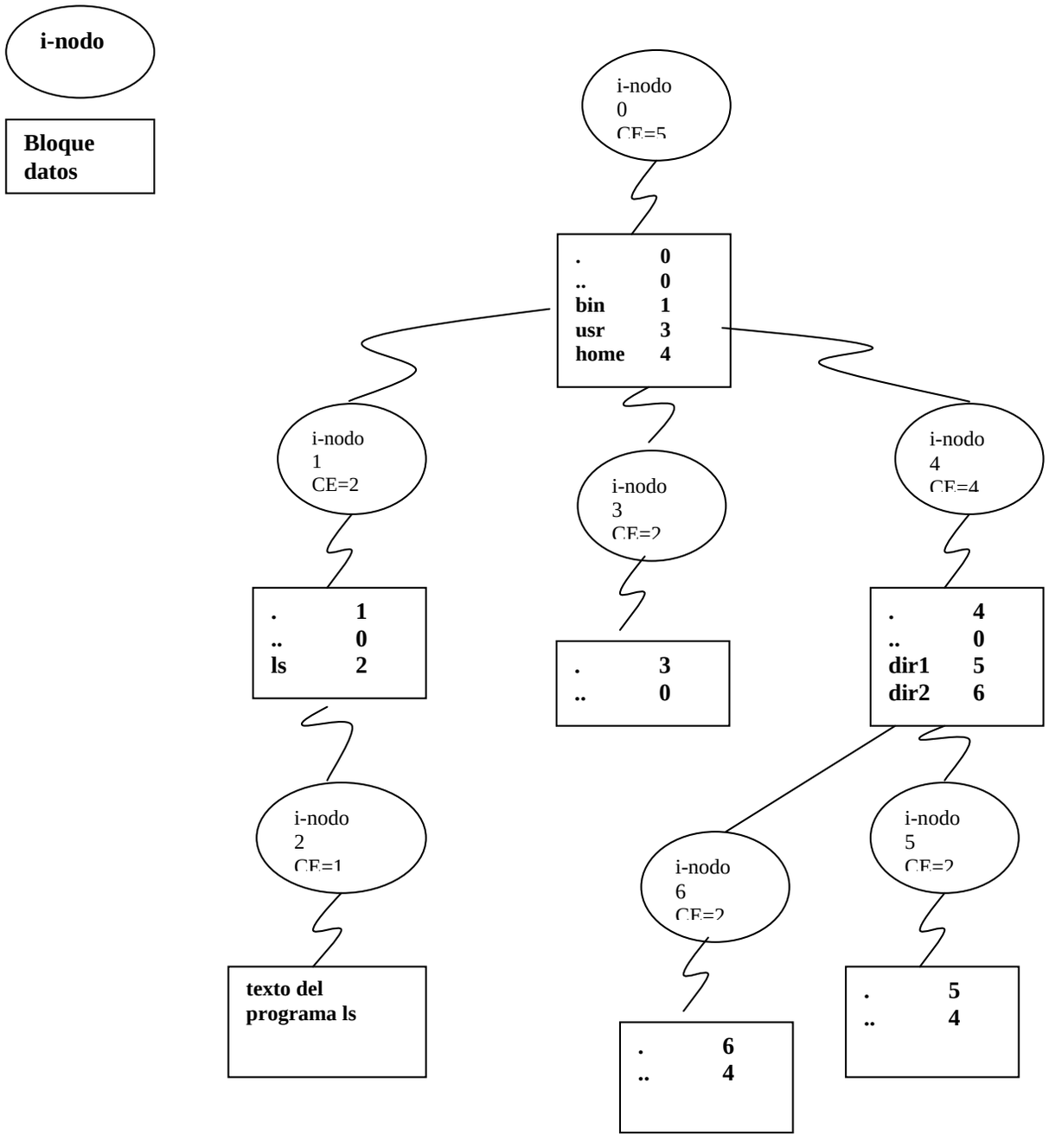
a) Dibujar el esquema de ficheros del sistema indicando en los i-nodos el valor del contador de enlaces y el contenido de todos los bloques de datos. **[1 punto]**

b) Aplicar las siguientes operaciones (asumiendo que el directorio de trabajo es /home/dir1), indicando como resultado de cada una qué ficheros y/o directorios se ven afectados, así como los contenidos de los directorios e i-nodos que se modifican **[2 puntos]**:

1. echo “HOLA” >/home/dir2/hola
2. ln -s /home/dir2 /home/dir2/copia
3. ln /bin/ls ./otro
4. rm ../dir1/otro
5. rm /home/dir2/copia

Solución:

a) Dibujo del esquema de directorios



b) Las operaciones producirían los siguientes efectos:

1. **echo "HOLA" > /home/dir2/hola**, se creará una entrada nueva en el inodo 6 que será **hola 7**, es decir, se ha creado el fichero hola en el i-nodo 7, cuyo contador de enlaces será 1 y apuntará a un bloque de datos cuyo contenido será **HOLA**
2. **ln –s /home/dir2 /home/dir2/copia**, con esto estamos creando un enlace simbólico del directorio **dir2** en un fichero que se llama **copia** y que está en el directorio **dir2**, por lo tanto aparecerá una nueva entrada en el directorio **dir 2** que será **copia 8**, ya

que al ser un enlace simbólico se habrá reservado un nuevo i-nodo, el i-nodo 8 cuyo contador de enlaces será uno y se habrá reservado un nuevo bloque de datos cuyo contenido será /**home/dir2**

3. **ln /bin/ls ./otro** en este caso estamos creando un enlace duro del fichero /**bin/ls** en el directorio /**home/dir1** ya que según el enunciado es el directorio de trabajo actual, lo que ocurrirá será que crearemos una nueva entrada en el directorio con el nombre **otro 2**, es decir, que se encuentra en el i-nodo 2 y habremos aumentado el contador de enlaces de /**bin/ls**
4. **rm ../dir1/otro** estaremos borrando la entrada **otro 2** de **dir 1** y bajaremos el contador de enlaces del **inodo 2** a 1
5. **rm /home/dir2/copia** en este caso estaremos borrando la entrada **copia 8** del directorio **dir 2** y además liberaremos el **i-nodo 8** y el bloque de datos al que estaba apuntando