SISTEMAS OPERATIVOS
2º Curso – Dobles Grados Informática
Ejercicios del Tema 4

1. Describa las estructuras de datos que se crean en la memoria del sistema operativo para trabajar con archivos indicando para que sirve cada una. Utiliza para ello como ejemplo el escenario en el que un programa abre el archivo *./dato.txt* solo para lectura.
2. ¿Qué diferencias habría en el caso de que el kernel fuese de Linux?
3. Si el proceso anterior realiza un `fork()` después de abrir el archivo, ¿cómo quedarían las estructuras de datos?
4. Explique el sistema de asignación de espacio en disco FAT y describe las ventajas e inconvenientes que presenta frente al método de asignación enlazado puro.
5. ¿Indicar cual es la estructura de un directorio en los sistemas de archivos: (a) V-FAT de Windows (b) ext2 de Linux?
6. Indicar como se implementa el concepto sesión de trabajo con un archivo de forma que el sistema permita a un proceso tener abiertas varias sesiones sobre un mismo archivo.
7. Sea el programa que se muestra a continuación y un archivo de texto, denominado *archivo_texto*, que contiene 30 caracteres "A". Se pide que;
   a) Dibujar los descriptores de las hebras que crea el kernel de Linux al ejecutar el programa, y sus principales contenidos.
   b) Dibujar las estructuras de datos relativas al manejo del archivo citado y sus principales contenidos.
   c) Indicar uno de los 2 posibles contenidos del archivo tras ejecutar una vez el citado programa.
   d) Indicar que diferencias habría en las estructuras de datos, si en el citado programa eliminamos el indicador `CLONE_FILES` de la llamada `clone()`.

Programa 1.-

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <linux/unistd.h>
#include <sys/syscall.h>
#include <errno.h>
#include <sched.h>
#include <fcntl.h>

int fd;

int thread(void *p) {

       int n;
       char buferH[10];
       close(1);
       dup(fd);
       n=read(fd, buferH, 10);
       printf("Hebra leyo: %s", buferH);
}

main() {
      unsigned char stack[4096];
      int i, m;
       char buferM[15];
```

```
        setbuf(stdout, NULL);
        fd=open("archivo_texto", O_RDWR);
        i = clone(thread, (void **) stack+2048, CLONE_VM|CLONE_FILES|CLONE_FS|
                          CLONE_THREAD|CLONE_SIGHAND , NULL);
         if (i == -1)
                 perror("clone");
         else
                 printf("clone retorna: %d", i);
        m=read(fd, buferM, 15);
        printf("Main ha leido: %s", buferM);
}
```

8. Sea el programa que se muestra a continuación y un archivo de texto, denominado *archivo_texto*, que contiene 30 caracteres "A". Se pide que:

a) Dibujar los descriptores de las hebras que crea el kernel de Linux al ejecutar el programa, y sus principales contenidos.

b) Dibujar las estructuras de datos relativas al manejo de archivos y sus principales contenidos cuando se alcanza el punto marcado con **(1)** en el programa.

c) Indicar el contenido del archivo *archivo_texto* tras ejecutar una vez el citado programa.

d) Indicar que diferencias habría en las estructuras de datos, si en el citado programa hubiésemos puesto el indicador `CLONE_FILES` de la llamada `clone()`.

Programa 1.-

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <sched.h>
#include <fcntl.h>

int fd;

int thread(void *p) {
   int fd2;
   symlink("archivo_texto", "nuevo_archivo");   (1)
   fd2=open("nuevo_archivo", O_RDWR);
   write(fd2, "BBBBB",5);
}

main() {
   unsigned char stack[4096];
   int i, m;
   char buferM[15];

   setbuf(stdout, NULL);
   fd=open("archivo_texto", O_RDWR);
   i= clone(thread, (void **) stack+2048, CLONE_VM|CLONE_FS|
                              CLONE_SIGHAND , NULL);
   if (i == -1)
        perror("clone");
   else
      printf("clone retorna: %d\n", i);
      m=read(fd, buferM, 15);
      printf("Main ha leido: %s\n", buferM);
}
```

9. Sea el siguiente programa:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

main() {
  char bufer1[] = "abcdef";
  char bufer2[] = "ghijkl";
  char bufer3[]= "";
```

```
int fd1, fd2;

fd1=open("archivo", O_RDWR|O_CREAT);
write(fd1, bufer1, 6);
fd2=open("archivo", O_WRONLY);
lseek(fd2, 6, SEEK_CUR);
write(fd2, bufer2, 6);
read(fd1, bufer3, 6);
printf("Leo de archivo: %s\n", bufer3);
close(fd1);
close(fd2);
    }
```

a) Dibujar las estructuras de datos en memoria que utiliza el subsistema de archivos para manipular *archivo* y qué información relevante contienen las mismas.
b) ¿Qué mostrará en pantalla la instrucción `printf` al imprimir `bufer3`?

10. En sistemas Unix, creamos una partición de disco para espacio de intercambio (*swapping*). Indicar que método de asignación de espacio es el más adecuado para esta partición. Comparar este esquema con el de Windows donde el intercambio se realizado sobre un archivo.