* Para la realización del presente examen se aconseja un tiempo de **2 horas.**

Teoría 1. Explique en detalle cómo se pasa una llamada al sistema operativo.

Teoría 2. Explique las diferencias, desde el punto de vista de imagen de memoria, de un proceso sin threads y con threads.

Ejercicio 1. Dado el programa que se muestra a continuación, responda a las cuestiones:

```
#include <stdio.h>
#include <stdlib.h>
main() {
   int pid,i, m=10;
   int tiempoinicial, tiempoactual;
   tiempoinicial = time(NULL);//time devuelve el tiempo actual en segundos
   tiempoactual = time(NULL) - tiempoinicial;
   printf("%d:Inicio del programa \n",tiempoactual );
   for(i=0; i<3; i++) {
       pid=fork();
       sleep(1);
       switch(pid) {
           case -1:
               perror("Error en la creación de procesos");
               exit(-1);
           case 0:
               m++;
               tiempoactual = time(NULL) - tiempoinicial;
               printf("%d:Hijo %d m=%d\n",tiempoactual, i, m);
               sleep(2);
               exit(0);
           default:
               tiempoactual = time(NULL) - tiempoinicial;
               printf("%d:Creado el proceso %d\n", tiempoactual, i);
               if( i%2 == 0 ) {
                   wait(NULL); //wait espera que finalice un hijo cualquiera
                   tiempoactual = time(NULL) - tiempoinicial;
                   printf("%d:Finalizó un proceso, valor de m=%d\n",
                              tiempoactual,m);
               } //fin if
       } //fin switch
   } //fin for
   wait(NULL);
   tiempoactual = time(NULL) - tiempoinicial;
   printf("%d:Finalizó un proceso, valor de m=%d",tiempoactual, m);
} //fin main
```

a) Escribir los mensajes que se escriben por pantalla y en qué instante, suponiendo que el mensaje de 'Inicio del programa' aparece en el instante 0.

b) ¿Cuántas variables 'm' se crean en memoria?

Ejercicio 2.

a) Escribir en código C una función llamada **"mi_cat"** que lea el contenido de un fichero (cuyo nombre se recibe por el primer parámetro) y lo escriba en un descriptor recibido por el segundo parámetro. Ayudarse del siguiente código:

```
void mi_cat(char *nombre_fichero, int fd_salida){

        int fd_entrada = open(nombre_fichero, O_RDONLY);

        if( fd_entrada < 0 ) {
            perror("Error al abrir el fichero");
            exit(-1);
        }

        /* RELLENAR POR EL ALUMNO */
            // LEER DE fd_entrada Y ESCRIBIR EN fd_salida
        /* FIN RELLENAR POR EL ALUMNO */

        if( close(fd_entrada) < 0 ) {
            perror("Error al cerrar el fichero");
            exit(-1);
        }
}
```

b) Escribir un programa en código C que ejecute:

mi_cat fichero_alumnos.txt | grep manuel

El proceso que ejecute 'mi_cat' debe enviar el contenido del archivo 'fichero_alumnos.txt' por la tubería, y el proceso que ejecute 'grep' coge los datos de la tubería y filtra sólo las líneas en las que aparece la palabra 'manuel'.

- Se deben utilizar procesos pesados comunicándolos mediante tuberías o pipes.
- El comando 'mi_cat' debe ejecutarse haciendo uso de la función del apartado a), en lugar de utilizar 'execvp' para ejecutarlo. En argv[1][0] se encuentra el nombre del fichero 'fichero_alumnos.xt'.
- Para el comando 'grep' suponer que en argv[2] se encuentra el comando con sus parámetros. De tal forma que para ejecutar este mandato grep, se debe utilizar la llamada al sistema 'execvp', de la forma: `execvp(argv[2][0], argv[2]);`

```
int main(int argc, char **argv[]){
      /* RELLENAR POR EL ALUMNO */
}
```