UNIVERSIDAD PÚBLICA DE NAVARRA

Grupo de Ingeniería Telemática

LABORATORIO DE TELEMÁTICA

Convocatoria ordinaria de Septiembre 1998

- Se permiten todo tipo de libros y apuntes
- Duración: 2 horas
- **Nota:** Responda en diferente hoja a las pregunta de Java respecto a las de UNIX

UNIX

Nota: Todos los programas compilan correctamente y en los códigos fuente se han omitido los `#include` para abreviar. Se supone que `%>` es el prompt de la Shell que se emplea.

Pregunta 1.- Tenemos el siguiente código fuente:

[comienzo de programa]
```
main(int num, char *cads[])
{
int i;

for (i=0;i<num;i++)
  printf("%s ", cads[i]);
printf("\n");
}
```
[final de programa]

Compilamos el programa:

```
%> gcc source.c -o ej1
```

Y lo ejecutamos de la siguiente forma:

```
%> ej1 -v 10 prueba
```

El programa se ejecuta correctamente. ¿Cuál es la salida que origina por pantalla?

Salida por pantalla:

ej1 –v 10 prueba

Pregunta 2.- El siguiente programa compila correctamente y se ejecuta sin fallos. Indique cuál es la salida por pantalla que origina y cuál es el contenido del fichero `prueba` al finalizar el proceso.

[comienzo de programa]
```
main()
{
int fd;

fd=open("prueba", O_CREAT|O_WRONLY, 0600);
fprintf(stderr, "Mensaje 1");
dup2(fd, STDERR_FILENO);
fprintf(stderr, "Mensaje 2\n");
}
```
[final de programa]

Salida por pantalla:

```
Mensaje 1
```

Contenido del fichero `prueba` *:*

```
Mensaje 2
```

Pregunta 3.- El siguiente programa se compila y ejecuta con normalidad. Indique cuál es su salida por pantalla.

[comienzo de programa]
```
main()
{
pid_t pid;

printf("Hola sin barra-n");
pid=fork();
printf("\n");
}
```
[final de programa]

Pregunta 4.- Indique la/s funcion/es que NO pueden crear ficheros en el sistema de ficheros:

a) creat b) pipe c) open
d) fopen e) msgget

b) y e)

Pregunta 5.- Expique las diferencias entre la llamada al sistema

```
ssize_t write(int fd, const void *buf, size_t count);
```

y la funcion de librería

```
size_t fwrite( void *ptr, size_t size, size_t nmemb,  FILE *stream);
```

`write` *es una llamada al sistema mientras que* `fwrite` *es una función de la librería estándar.* `write` *actúa sobre un descriptor de fichero mientras que* `fwrite` *lo hace sobre un stream.* `write` *vuelca en el fichero una cantidad de bytes especificado de una zona de memoria,* `fwrite` *vuelca en un stream una cantidad de estructuras de un tamaño especificado cada una que están de forma consecutiva en una zona de memoria.*

Pregunta 6.- El siguiente programa compila correctamente:

[inicio de programa]
```
main()
{

 if (signal(SIGKILL, SIG_IGN)==SIG_ERR)
       fprintf(stderr, "Error\n");
 else
       printf("Ok, seguimos adelante\n");
}
```
[final de programa]

Al ejecutarlo el resultado por pantalla es:

```
Error
```

Explique por qué.

Porque SIGKILL es una señal que no puede ser ignorada, lo cual se pretende imponer con ese empleo de la llamada `signal`.

Pregunta 7.- ¿Qué valores puede devolver la llamada al sistema `read()` y qué significado tienen?

Devuelve un valor entero estrictamente positivo cuando se ejecuta con éxito. Este valor es el número de bytes leídos. Devuelve 0 para indicar el fin de fichero y –1 para indicar un error.

Pregunta 8- Indique alguna función estándar con la que se pueda cambiar el modo en que está configurado un stream.

setvbuf

Pregunta 9.- El siguiente programa se dedica a ir contando segundo a segundo, haciendo aparecer el contador en la pantalla:

[comienzo de programa]
```
main()
{
int cont,fd;

printf("Segundo:\n");
for (cont=1; cont<21; cont++)
{
  printf("%i\n", cont);
  sleep(1);
}
exit(0);
}
```
[final de programa]

Lo compilamos, el ejecutable se llama ej9.
Lo ejecutamos de la siguiente forma:

```
%> ej9
```

y vemos cómo por pantalla sale la cuenta de segundos tal y como esperábamos. Hacemos ahora:

```
%> ej9 > salida &
```

para ir guardando esta cuenta en un fichero y mientras hacer otra cosa
Nos aparece de nuevo el prompt de la Shell. Hacemos `ls` y encontramos el fichero `prueba`, sin embargo, al hacer `cat fichero`, vemos que está vacío, esperamos un par de segundos y sigue vacío. Si esperamos a que el programa termine encontramos el fichero `prueba` que contiene toda la cuenta. Explique a qué se debe que suceda esto.

Se debe a que al redireccionar la salida estándar a un fichero pasa a tener buffer completo, con lo que lo que se envíe a ella no se escribe en el fichero hasta que se llene el buffer o se haga un fflush sobre su stream. En este caso al terminar el programa cierra todos los streams, proceso durante el cual se hace un fflush de ellos para volcar lo que quedara en los buffers; en ese momento aparece la cuenta en el fichero.

Pregunta 10.- Indique cuál es la salida por pantalla de este programa (compila correctamente y el `fork()` se ejecuta sin errores). El ejecutable `ej1` se supone que es el de la pregunta 1.

[comienzo de programa]
```
main()
{
  pid_t pid;
```

```
        pid=fork();

        if (pid<0)
              exit(-1);

        if (pid>0)
        {
              execlp("ej1", "ej1", "op1", "op2", NULL);
              printf("Fin de ej1\n");
        }
        else
        {
              execlp("ej1", "ej1", "op2", "op4", NULL);
              printf("Fin 2 de ej1\n");
        }
      }
```
[final de programa]

Una posible salida por pantalla es:

```
ej1 op1 op2
ej1 op2 op4
```

JAVA (Acuerdese de utilizar una hoja nueva)

Listado para las cuestiones 11,12,13,14 y 15

```
public class punto {
  int x,y;

  punto() {
    x=0;
    y=0;
  }

  punto(int x, int y) {
    this.x=x;
    this.y=y;
  }
}
```

Pregunta 11.- ¿Cómo se construye un objeto de tipo punto para representar al punto x=2,y=5?

a) `punto p = new punto[2][5];`
b) `punto p = punto[2][5];`
c) `punto p = new punto(2,5);`
d) `puntp p = punto(2,5);`

Pregunta 12.- Programe un tercer constructor para la clase punto que tome como parámetro un objeto de tipo punto y que construya otro punto igual al que se le pase.

```
  punto(punto p) {
    this.x=p.x;
    this.y=p.y;
  }
```

Pregunta 13.- Programe dos metodos para la clase punto que calculen la distancia entre dos puntos p y q. Un metodo estático que se utilice haciendo `distancia(p,q)` y otro que se invoque sobre uno de los puntos `p.distancia(q)` . La distancia que devuelven sera en ambos casos un

`double`. (Nota: para hacer raices cuadradas se puede utilizar el metodo `Math.sqrt(double)` que devuelve un `double`)

```
public static double distancia(punto a,punto b) {
    double modulo;
    int dx,dy;

    dx=a.x-b.x;
    dy=a.y-b.y;
    modulo=Math.sqrt(dx*dx+dy*dy);
    return (modulo);
  }

  public double distancia(punto p) {
    return distancia(this,p);
  }
```

Pregunta 14.- Programe un método `toString()` que devuelva una cadena de texto con la representación del punto en formato (x,y). Por ejemplo `p.toString()` devuelva la cadena `"(2,5)"`

```
public String toString() {
    String s;
    s="("+x+","+y+")";
    return s;
  }
```

Pregunta 15.- Suponga que en un método tenemos la siguiente declaración.

```
punto[] pts = new punto[20];
```

Escriba como introduciría el punto x=6,y=2 en la posición 5 del array.

```
pts[5] = new punto(6,2);
```

Listado para las cuestiones 16,17

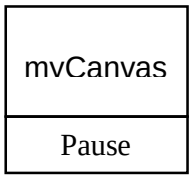
```
import java.awt.*;

public class myCanvas extends Canvas implements Runnable {
  int x,y,vx,vy;
  int w,h;
  Thread t;

  myCanvas() {
    x=20;y=10;vx=12;vy=7;
    w=this.size().width;
    h=this.size().height;
    this.start();
  }

  myCanvas(int x,int y,int vx,int vy) {
    this.x=x;
    this.y=y;
    this.vx=vx;
    this.vy=vy;
    w=this.size().width;
    h=this.size().height;
    this.start();
```

```
  }

  public void start() {
    t = new Thread(this);
    t.start();
  }

  public void paint(Graphics g) {
    w=this.size().width;
    h=this.size().height;
    g.drawOval(x-10,y-10,20,20);
  }

  public void run() {
    while (true) {
      try {
    Thread.sleep(100);
      } catch (InterruptedException e) {};
      if (x>w||x<0) vx=-vx;
      if (y>h||y<0) vy=-vy;
      x=x+vx;
      y=y+vy;
      repaint();
    }
  }
}
```

Pregunta 16.- Explique brevemente el funcionamiento de la clase `myCanvas` definida en el listado anterior. Qué dibuja en pantalla y cómo lo hace.

Va dibujando un circulo que rebota en los bordes de la pantalla. Los constructores permiten especificar la posicion inicial y la velocidad inicial, o dejar que se utilizen valores por defecto. Al construir el objeto se llama a start() que crea un hilo para ejecutar run(). El hilo calcula las nuevas coordenadas y manda redibujar el Canvas cada decima de segundo (100ms).

Pregunta 17.- Escriba el método `init()` de un Applet que contenga un objeto `myCanvas` con un botón debajo con la etiqueta "Pause", como se ve en la figura. (Nota: sólo el metodo `init()` no hace falta hacer funcionar el botón)




```
myCanvas c;

  public void init() {
    setLayout(new BorderLayout());
    add("Center",c=new myCanvas());
    add("South",new Button("Pause"));
  }
```

Pregunta 18.- ¿Qué diferencia fundamental hay entre un Canvas y un Panel?

Listado para las cuestiones 19,20

```
public class helloworld {
```

```
  public static void main(String[] args) {
    System.out.println("Hello World !!!");

  }

}
```

Pregunta 19.- Modifique el listado anterior para que el texto "Hello world!!!" aparezca en una ventana en lugar de ir a la salida estandar. (Nota: sin preocuparse de como cerrar luego la ventana).

```
import java.awt.*;

public class helloworld {

  public static void main(String[] args) {
    Frame w;

    w=new Frame("HelloWorld");
    w.resize(200,100);
    w.setLayout(new GridLayout(1,1));
    w.add(new Label("Hello World !!!"));
    w.show();
  }

}
```

Pregunta 20.- Una vez modificada se compila la clase anterior haciendo `javac helloworld.java`. Pero ¿como se ejecuta?

a) `java helloworld`
b) `java helloworld.class`
c) `appletviewer helloworld.class`
d) `appletviewer helloworld.html`