



PREGUNTAS CORTAS DE INFORMÁTICA I

Nombre y Apellidos/ Izena eta Deiturak:			
DNI/NAN:	Curso/Ikasurte: 2º	Grupo/Talde:	

1. Indica al menos tres ventajas de los Sistemas Operativos con interfaz gráfica.

Son Intuitivos, Fáciles de usar.

Son Atractivos.

Son capaces de ofrecer más información en menos espacio.

Son capaces de destacar información crítica.

2. Indica al menos tres ventajas de los Sistemas Operativos con interfaz orientado al carácter.

Más fáciles de programar.

Ocupan menos bytes.

Suelen tener menos bugs.

Son más rápidos ejecutándose.

3. Por defecto, ¿a qué usuarios corresponden en Linux los siguientes prompts: "\$" y "#"?

"\$": Usuario no root

"#": Usuario root

4. ¿Qué diferencia existe en cuanto al resultado obtenido con los comandos `cat /etc/joe/terminfo` y `cat < /etc/joe/terminfo`?

Ninguna.

5. ¿Qué diferencia existe en cuanto al resultado obtenido con los comandos `cat /etc/joe/terminfo > prueba.txt` y `cat < /etc/joe/terminfo > prueba.txt`?

Ninguna.

6. ¿Qué diferencia existe en cuanto al resultado obtenido con los comandos `ls a.txt | head` y `ls a.txt | tail`?

El primero muestra las primeras líneas del fichero, el segundo las últimas.

7. Crear un único comando que sea capaz de llevar a un fichero la lista de todos los usuarios ordenada alfabéticamente.

cat /etc/passwd | sort > sorted.users

8. Crear un único comando que busque por todo el sistema de archivos todos los ficheros cuyo nombre comience por la subcadena "pro1".

find / -name pro1*

9. ¿Cuáles de estas tres combinaciones de teclas tienen el mismo efecto: "<ctrl.->", "<ctrl.-z>", "<ctrl.-d>". Describelos

Todas tienen efectos distintos.

<ctrl-c> envía una señal a un proceso, de manera que si ésta no está capturada, el proceso muere.

<ctrl.-z> permite suspender un proceso.

<ctrl.-d> indica un "fin de fichero" o "end of file".

10. Crear un script que solicite al usuario tres nombres de archivo, indicando al usuario que los dos primeros deben existir y el tercero no, y genere el tercer archivo a partir de la concatenación de los dos primeros

```
echo Introduce el nombre de archivo 1 (existente)

read archivo1

echo Introduce el nombre de archivo 2 (existente)

read archivo2

echo Introduce el nombre de archivo 3 (inexistente)

read archivo3

echo Ejecutando: cat $archivo1 $archivo2 > $archivo3

cat $archivo1 $archivo2 > $archivo3

echo ok
```

11. Crear un script que pide un nombre de carpeta inexistente, crea esa carpeta, y dentro de esa carpeta crea una segunda carpeta del mismo nombre, y dentro de esa una tercera con el mismo nombre

```
echo Introduce el nombre de carpeta
read nomCarpeta
mkdir $nomCarpeta
cd $nomCarpeta
mkdir $nomCarpeta
cd $nomCarpeta
mkdir $nomCarpeta
```

12. ¿Qué es un shell? ¿Entre qué dos elementos constituye un interfaz?

Es el interfaz entre los usuarios del sistema y el sistema operativo.

Es un programa del sistema, y por tanto, interfaz entre los usuarios y el sistema operativo.

13. Describe los pasos que se siguen desde que el shell recibe la cadena de caracteres tecleada por el usuario hasta que se muestra la respuesta generada por pantalla.

El shell recibe una cadena de caracteres que el usuario teclea.

El shell analiza la cadena de caracteres tecleada por el usuario y extrae de ella el nombre del (los) comando(s) introducidos.

El shell busca un programa de igual nombre al comando extraído y crea un proceso para que se encargue de la petición, pasándole los posibles parámetros recibidos.

El shell espera la finalización del proceso, y al recibir la respuesta, la muestra por pantalla.

14. Enumera al menos 4 shells distintos de Linux, contando algunas características de alguno de ellos.

Shells: Bourne, C, Korn y Bash.

El shell Bash es el predeterminado de Linux y se encuentra en /bin/bash. Proporciona optimizaciones como un historial de órdenes y la capacidad de edición y/o terminación de las mismas. Está incluido en todas las distribuciones de Linux.

15. ¿Qué es el entorno del terminal? ¿Qué elemento del sistema operativo se encarga de su gestión?

El entorno del terminal es el conjunto de informaciones relativas a la terminal de usuario que maneja el sistema operativo. El elemento encargado de la gestión de esta información es el controlador de dispositivos.

16. ¿Cuál es el contenido de las variables de shell más habituales?

HOME: Nombre completo de la ruta del directorio de usuario.

SHELL: Nombre del shell actual.

MAIL: Nombre completo de la ruta del buzón de correo.

LOGNAME: Nombre de entrada del usuario al sistema.

PATH: Directorios que el shell revisa en busca de órdenes.

17. ¿Cuáles son los tres comodines más utilizados en Linux? ¿Qué significa cada uno?

"*": Se sustituye por cualquier conjunto de caracteres.

"?": Se sustituye por un carácter.

[...]: Se sustituye por el rango de caracteres especificado en la serie.

18. ¿Para qué se utilizan los pipes o tuberías?

Los pipes o tuberías permiten la conexión o comunicación de dos o más procesos (y por tanto de dos o más comandos) de manera que la salida del primero de ellos se considerará como entrada para el segundo.

19. Enumera los pasos necesarios para escribir y ejecutar un programa de shell. ¿Por qué no es necesario compilarlo?

Escribir el código del programa en un editor.

Salvar el código del programa en un fichero.

Cambiar los permisos del fichero para que éste sea ejecutable.

Ejecutar el fichero.

No es necesario compilarlo porque contiene comandos directamente comprensibles para el sistema operativo, y no código en un lenguaje de programación.

20. ¿Cuál es la diferencia entre escribir en un programa de script o en línea de comandos "<var>=<valor>", en vez de "<var> = <valor>"?

En los programas de script, no es posible dejar blancos entre <var> y el signo "=", y entre el propio signo "=" y <valor>. Simplemente, el programa de script no funcionaría. Lo mismo ocurre en línea de comandos.

21. Escribe el código de un script que solicite la introducción del nombre del usuario por pantalla y a continuación muestre el mismo.

echo introduce tu nombre

read nombre.

echo tu nombre es

echo \$nombre

22. Escribe el código de un script que permita compilar todos aquellos ficheros de extensión ".c" que se encuentren en un directorio.

for i in *.c

do

gcc \$i -o \$i.exe

done

23. ¿Qué es la programación?

Es la escritura de secuencias de instrucciones encaminadas a resolver, mediante su ejecución, cierta problemática.

24. ¿Qué es el lenguaje máquina? ¿Cuál es su principal ventaja? ¿Y su desventaja?

Es el lenguaje formado por series de unos y ceros. Su ventaja principal radica en que es el único lenguaje que comprende el ordenador directamente. Su desventaja es la tediosidad que conlleva la escritura de programas en el mismo.

25. ¿Cuál es la diferencia fundamental entre un compilador y un intérprete?

La diferencia principal radica en que los compiladores traducen o compilan el programa completo, generando, si todo es correcto, un código objeto que puede ser almacenado. Este código suele ser más óptimo y eficiente que el de los intérpretes. El intérprete, por el contrario, va traduciendo instrucción a instrucción, generando una realimentación inmediata, y facilitando así la detección y corrección de errores.

26. ¿Qué es una señal? ¿Cuál es el tratamiento por defecto de una señal en el caso del proceso receptor?

Es una indicación generada por parte de un proceso para indicar a otro proceso (o al sistema) que se ha cumplido cierta condición. Cuando un proceso recibe una señal, el proceso receptor "se muere".

27. ¿En qué punto de la ejecución de un programa es más conveniente capturar una señal?

La captura de una señal debe hacerse siempre lo antes posible, puesto que las señales constituyen un método de comunicación asíncrono, que podrían abortar la ejecución de un proceso en cualquier instante.

28. Enumera y describe al menos tres de las señales más utilizadas en Linux.

SIGINT: es la señal que se genera por pulsación de las teclas <ctrl.-c>.

SIGKILL: es la señal que garantiza la muerte del proceso receptor, al no poder ser capturada.

SIGTERM: es la señal que por defecto envía el comando kill.

29. ¿Cuál es el sentido de la función pause() en Linux? ¿Hay alguna excepción que varíe su comportamiento?

La función pause() bloquea al proceso que la ejecuta hasta que reciba una señal. No obstante, hay una excepción: si la señal estaba ignorada previamente, el proceso que ejecute la pause() quedará bloqueado aunque llegue la señal.

30. Escribe el código de un programa en C que capture la señal SIGFPE (generada cuando se produce una división por cero) mostrando un mensaje por pantalla cada vez que llegue alguna de ellas. Compíllalo en "vega" y anota cualquier posible incidencia en su ejecución.

```
#include <stdio.h>

void trapper(int);

int main (void)
{
    int a;

    signal(8,trapper);

    for (;;) {
        a=a/0;
    }
}

void trapper(int sig)
{
    signal(sig,trapper);

    printf("Recibida la señal: %d\n",sig);
}
```

