Trabajo Práctico N°1

Materia: SEMINARIO LENGUAJES II

Docentes: Ing. Enrique De Rezende, Gustavo Yauny

Grupo:

Integrantes: Barreto Florencia, Bazyluk Federico, Cano Jorge, Méndez Matias.

Tema: Conceptos / Introducción – Linux y Lenguaje C / C++

PARTE I: CONCEPTOS SISTEMAS OPERATIVOS LINUX

- Defina, explique resumidamente, y de ejemplos de los siguientes conceptos, para diferentes distribuciones del S.O. Linux: (al menos 5)

1) ¿Qué es un Kernel ?
    - Se denomina Kernel al núcleo, corazón o motor que hace funcionar a un Sistema Operativo, el mismo se encarga de la interacción del hardware con el software, es decir, administra el uso de la memoria, gestiona los procesos en ejecución, controla periféricos, entre otras tareas de bajo nivel. Es imposible tener un sistema operativo funcional sin un kernel, un ejemplo claro es GNU/Linux.

2) ¿Qué es una Distribución ? ejemplos
    - Una distribución GNU/Linux o llamado de otro modo como distro no es más que un software basado en el kernel de Linux con un gestor de paquetes específico o heredado, un conjunto de aplicaciones pre-instaladas y un destino de uso determinado a ciertos usuarios. Un ejemplo claro sería Debian (el cual es padre de varias distros) que dio origen a Ubuntu, Kali Linux, BackTrack (ya obsoleta), otra distro muy distinta sería ArchLinux, Manjaro (basada en Arch), entre otros.

3) Tipos de Sistemas de Archivos en Linux, explique, ejemplifique.
    - Un sistema de archivos en una capa bajo el S.O que maneja el posicionamiento de los ficheros y/o directorios que se almacenan en el disco.
      Los sistemas operativos basados en linux ofrecen varios tipos de sistemas de archivos, algunos muy populares o utilizados y otros ya discontinuados:

        # ext: Un viejo sistema de archivos ya obsoleto por sus limitantes.
        # ext2: Fué el primer sistema de archivos en poder manejar 2Tb de datos.
        # ext3: Una mera evolución de ext2 con algunos cambios.

        # ext4: El, hasta el momento, sistema de archivos más usado y conocido por por su velocidad; es por eso que al instalar cualquier distro linux, este es el

sistema de archivos por defecto seleccionado.
jfs: Diseñado por IBM, este sistema de archivos funciona bien de cualquier tamaño, pero presenta fallas después de un tiempo de uso.
xfs: Es antiguo y funciona lento con archivos de pequeño tamaño.
btrfs: Creado por la empresa Oracle, no es tan estable como el sistema “ext”, pero si es necesario utilizarlo, cumple con un rendimiento muy bueno.

4) Conceptos de Particiones y particiones SWAP
   - Se denomina particiones a las divisiones del disco o volúmenes del mismo creados para un uso determinado o el almacenamiento de ciertos directorios y ficheros, como por ejemplo (en el caso de Sistemas Linux) directorio Raíz (el cual almacena el kernel, binarios del sistema, ficheros .config de programas, Logs/registros, archivos temporales, entre otros), directorios de usuarios, directorio Root (superusuario); o por ejemplo la creacion de volumenes que almacenan el gestor de arranque, sino también particiones que funcionan como memoria virtual, etc. Existen 3 tipos de particiones: Primarias, Extendidas/Secundarias y Lógicas.
     En Linux se considera partición SWAP al volumen dentro del disco que será utilizado como memoria virtual, es decir que si existiese el caso de que la memoria principal se llene/consuma, dicha partición del disco se utilizara como si de una memoria RAM se tratara.

5) Gestores de Arranque, principios, objetivos etc.
   - El gestor de arranque es un componente de software que ayuda al hardware y al firmware del equipo a iniciar el S.O, además permite la elección de distintos S.O instalados. Normalmente los gestores de arranque forman parte del cargador de arranque como por ejemplo GRUB (el más conocido) o GRUB2, entre otros. Sin embargo el gestor de arranque puede ser un programa independiente y funcionar de dicha forma (un ejemplo sería “fallback.efi”), además puede ejecutar otros programas que testeen la memoria o verifique el hardware del equipo al iniciar el sistema.

6) Comandos en Linux (bajo terminal) nombre y ejemplifique unos 10 comandos.
   - Mediante un emulador de terminal o CLI (Command-line Interface / Interfaz de Línea de Comando) podemos realizar varias acciones a

través de la ejecución de comandos ingresados por el usuario, algunos de ellos son:

> **whoami** -> Imprime en terminal el nombre de usuario actual.
> **mkdir** -> Crea un directorio.
> **rm** -> Elimina un directorio.
> **cd** -> Ingresa a un directorio.
> **cp** -> Copia un directorio.
> **mv** -> Mueve un directorio de una ruta a otra.
> **ls** -> Lista el contenido de un directorio.
> **touch** -> Crea un fichero.
> **echo** -> Imprime un texto en terminal, ingresado por el usuario.
> **pwd** -> Imprime la ruta del directorio actual en la que se encuentra el usuario.
> **sudo** - Ejecuta comandos en modo usuario "root".
> **tail** -> Muestra las últimas 10 líneas de un fichero.
> **head** -> Muestra las primeras 10 líneas de un fichero.
> **man** -> Imprime el manual (si lo tiene) de un comando solicitado.
> **grep** -> Imprime líneas de un fichero que coincidan con los argumentos ingresados.
> **more** -> Imprime el contenido de un fichero de forma más cómoda para su lectura.
> **file** -> Imprime el tipo de archivo de un fichero solicitado.
> **mount** -> Monta una partición del sistema.
> **du** -> Imprime el tamaño de un directorio o fichero.
> **which** -> Imprime la ruta de ubicación de un comando determinado.
> **cat** -> Imprime el contenido de un fichero.

7) Explique servicios de Red en linux y su compatibilidad con Windows.

La Red es el medio por el cual el servidor puede comunicarse con los usuarios y con otras máquinas, tanto servidores como clientes, permitiendo el intercambio masivo de información entre ordenadores.
De acuerdo con la planificación efectuada, la empresa debia contar con una infraestructura adecuada para el intercambio de datos.
El conjunto de protocolos TCP/IP, creados y desarrollados en UNIX y administrados por entidades internacionales de estándares, son continuamente reelaborados por Microsoft para impedir la interoperabilidad entre los sistemas operativos.
Microsoft Corporation se apropió e hizo extensiones de un protocolo de seguridad de red público y abierto, Kerberos, que fue desarrollado en el

MIT y se puso a disposición de toda la comunidad informática libre de coste.
Está extendida la *sospecha*, entre los diseñadores de Kerberos y otros expertos, de que las extensiones que Microsoft introdujo a su implementación de Kerberos en Windows 2000 no tenían otro propósito que hacer los productos de los competidores incompatibles con las estaciones de trabajo de Microsoft, para hacer que las empresas adoptaran servidores Microsoft en lugar de UNIX/Linux.

El producto "Directorio Activo" de Microsoft fue el centro de los planes de la compañía para competir con los entornos de servicios de directorio independientes de la arquitectura, como los estándar abiertos LDAP y Samba o el "Netware" de Novell, entre otros.

https://linuxcomputronic.blogspot.com/

8) Gestores de Paquetes, instalación / actualización de software.- defina, explique, ejemplifique.
    - En Linux, los gestores de paquetes son programas diseñados para administrar, instalar, actualizar, desinstalar, verificar, etc; los distintos conjuntos de softwares o componentes de diferentes tipos, es decir que, los gestores de paquetes vienen a facilitar y mantener un control de todas las instalaciones y actualizaciones (entre otras tareas de gestión de paquetes) de programas de forma automatizada, ya que no es tan simple hacerlo de forma manual. Este concepto no es propio de linux, hasta lenguajes de programación utilizan gestores de paquetes, un ejemplo es python con PIP, o javascript con NPM.
      Existen distintos gestores de paquetes (de hecho es una de las grandes diferencias entre las distribuciones [además de sus filosofías]), algunos de ellos pueden ser:
        - Pacman - Gestor de paquetes de Arch Linux (una distribución para usuarios avanzados) y Manjaro (distribución basada en Arch Linux, para los usuarios menos experimentados). EJ de instalación de paquete: **sudo pacman -Sy <paquete>**

        - APT - Gestor de paquetes que pertenece a debian, ubuntu (y todos sus sabores), kali linux, entre otras distros basadas en debian, este gestor administra paquetes “dpkg”. EJ de instalación de paquete: **sudo apt install <paquete>**

- RPM - Es un gestor de paquetes creado por RedHat, del mismo se crearon otros sistemas que lo mejoran o aportan a él. EJ de instalación de paquete: **sudo rpm -i <paquete>**

- YUM - Es utilizado en S.O's como Fedora y CentOS. EJ de instalación de paquete: **sudo yum localinstall <paquete>**

PARTE II: LENGUAJE DE PROGRAMACIÓN C y C++ (para sistema operativos Linux y Windows)

1) defina lenguajes de programación, tipos, clasificación, ejemplos.,

Lenguajes de programación:

Es el medio por el cual se comunica a cualquier dispositivo que lo requiera, una serie de acciones o instrucciones con el fin de lograr un objetivo.

Esto se produce con la escritura de un código, que el dispositivo interpreta y que da como resultado una acción o un resultado deseado.

Tipos de lenguajes de programación:

De acuerdo a su finalidad y herramientas en las que se usa, hay tres clases de lenguajes de programación:

- Lenguaje máquina: Es el más primitivo de los códigos y se basa en la numeración binaria, todo en 0 y 1. Este lenguaje es utilizado directamente por máquinas o computadoras.
- Lenguajes de programación de bajo nivel : Es un lenguaje un poco más fácil de interpretar, pero puede variar de acuerdo a la máquina o computadora que se esté programando.
- Lenguajes de programación de alto nivel: En esta categoría se encuentran los más utilizados. Se usan palabras del inglés lo cual facilita que una persona pueda intervenir más fácil que en los dos anteriores.
  Según su generación y orden cronológico, los lenguajes de programación de alto nivel se clasifican en:

  - Primera generación: el lenguaje máquina y el ensamblador.
  - Segunda generación: aquí encontramos los primeros lenguajes de programación de alto nivel, ejemplos de ellos son FORTRAN, COBOL.
  - Tercera generación: en esta generación encontramos los lenguajes de programación de alto nivel imperativo, pero mucho más utilizados y vigentes en la actualidad (ALGOL 8, PL/I, PASCAL, MODULA).
  - Cuarta generación: más cercanos a la época actual, es común encontrarlos en aplicaciones de gestión y manejo de bases de datos (NATURAL, SQL).
  - Quinta generación: estos son los más avanzados y fueron pensados para la inteligencia artificial y para el procesamiento de lenguajes naturales (LISP, PROLOG).

Ejemplos de lenguajes de programación:

- Java: Es el más actualizado actualmente y esto se debe a su simplicidad y legibilidad. No en vano, más de 9 millones de usuarios lo usan y está presente en miles de millones de dispositivos, lo que significa un 15% del total en el mundo frente a otros lenguajes.

Se ha mantenido a lo largo del siglo XXI en las primeras posiciones, lo cual da una idea de lo importante que ha sido este lenguaje de programación en lo que utilizamos hoy en día.
También es una de las habilidades más requeridas entre desarrolladores, tanto es así que es de el más importante skill que buscan las principales empresas de software y tecnología.
Es utilizado en diferentes segmentos, tales como aplicaciones móviles, herramientas para aprendizaje, hojas de cálculo, entre otras.

- Lenguaje de programación C: Surgió en los años 70 y tenía un nombre que no cambió mucho: «B».
  Con un alto porcentaje de uso al igual que java, 12%, es el segundo lenguaje más usado a nivel mundial. Su uso también es muy variado y se puede ejecutar en la mayoría de los sistemas operativos.
  Es comúnmente utilizado en aplicaciones de escritorio.

- Python: Un lenguaje de programación multiplataforma y multiparadigma, que también tiene un propósito general. Esto significa que soporta la orientación a objetos, la programación imperativa y funcional.
  Su sencillez, legibilidad y similitud con el idioma inglés lo convierten en un gran lenguaje, ideal para principiantes.

- Visual Basic. NET: Ha tenido una rápida evolución en número de usuarios en los últimos años. Conocido por ser una herramienta mucho más amigable, que no exige tanto conocimiento como, por ejemplo, el C#.
  Por su sencillez, es bastante utilizado para herramientas de automatización de procesos y aplicaciones web,

2) defina el punto anterior para Lenguaje C y C++

Lenguaje C:
Es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear softwares de sistemas y aplicaciones.
Se trata de un lenguaje de tipos de datos estáticos, débilmente tipado, de medio nivel, que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a

su vez, dispone de construcciones del lenguaje que permiten un control a bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La compilación de un programa C se realiza en varias fases que normalmente son automatizadas y ocultadas por los entornos de desarrollo:

Preprocesado consiste en modificar el código fuente en C según una serie de instrucciones (denominadas directivas de preprocesado) simplificando de esta forma el trabajo del compilador. Por ejemplo, una de las acciones más importantes es la modificación de las inclusiones (#include) por las declaraciones reales existentes en el archivo indicado.
Compilación que genera el código objeto a partir del código ya preprocesado.
Enlazado que une los códigos objeto de los distintos módulos y bibliotecas externas (como las bibliotecas del sistema) con el código objeto generado en el paso anterior para generar el programa ejecutable final.

Ejemplo:

```c
/* Suma de n números */

#include <stdio.h>
int main() {
    int num=0,suma=0;

    do {
        suma=suma+num;
        printf("un número: ");
        scanf("%d",&num);
    } while(num>=0);
    printf("suma es: %d",suma);
    return 0;
}
```

Lenguaje C++:
Es un lenguaje de programación diseñado en 1979 por Bjarne Stroustrup. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación

orientada a objetos. Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Características de C++

- Su sintaxis es heredada del lenguaje C.
- Programa orientado a objetos (POO).
- Permite la agrupación de instrucciones.
- Lenguaje muy didáctico, con este lenguaje puedes aprender muchos otros lenguajes con gran facilidad.
- Es portátil y tiene un gran número de compiladores en diferentes plataformas y sistemas operativos.
- Permite la separación de un programa en módulos que admiten compilación independiente.
- Es un lenguaje de alto nivel.

Ejemplo Hola mundo escrito en C++

```cpp
/* Esta cabecera permite usar los objetos que encapsulan los descriptores stdout
y stdin: cout(<<) y cin(>>)*/
#include <iostream>

using namespace std;

int main()
{
    cout << "Hola mundo" << endl;
    return 0;
}
```

3) ventajas y desventajas de C.
   Ventajas:
   ➔ Alto nivel de desempeño: El lenguaje C es muy eficiente para hacer llamadas directas al sistema operativo. Ofrece variedad de parámetros de optimización.

Su sistema inteligente impide generar operaciones sin sentido, tiene uso eficiente de la memoria y cuenta con funciones y variables estáticas, lo cual permite que los punteros direccionen todo el uso hacia la memoria.

- ➔ Uso de lenguaje eficiente: Utiliza lenguaje compilado y se acopla de forma efectiva con el lenguaje ensamblador. Es el que mejor aprovecha la CPU de la máquina.
  El lenguaje C es simple, flexible y con múltiples estilos. Tiene añadida funciones matemáticas, manejo de archivos, bibliotecas, etc.
  Crea grupos pequeños de palabras clave, lo cual ayuda a no memorizar códigos.
- ➔ Multiplataforma: El lenguaje C puede ser ejecutado en cualquier tipo de software o hardware. Gracias a esto, se pueden desarrollar aplicaciones compatibles con sistemas operativos como Linux, Mac, Windows.
- ➔ Estructura de datos: El lenguaje C permite que gracias a la sentencia struct, se acceda a formar una variable de registros. Un ejemplo de esto es la variable “fecha” que está compuesta por tres registros tipo int, los cuales son día, mes y año.
  A su vez, la comunidad está constantemente mejorando el núcleo del sistema, lo cual te permite desarrollar aplicaciones con múltiples patrones de diseño.
- ➔ Base de datos: Uno de los beneficios más importantes en la comparativa de ventajas y desventajas del lenguaje C es la buena gestión que garantiza el intercambio, consulta, almacenamiento y actualización de los datos.
  Utiliza un sistema de distribuidos que permite desarrollar aplicaciones en la nube y es compatible con cualquier hardware. Puedes hacer acciones escalables como el almacenamiento de datos.

Desventajas:

- ★ Lenguaje incompleto: El lenguaje C no tiene los suficientes operadores para hacer más abstracta la traducción del sistema. No tiene un lenguaje visual, por lo que impide que se pueda deducir intuitivamente, a diferencia de otros programas como Visual Basic y Python.

- ★ Estructura muy cerrada : El lenguaje C sufre de una encapsulación de las funciones anidadas, ya que para el uso correcto de estas se necesita de extensiones.
  A su vez, no cuenta con instrucciones de entrada ni de salida, tampoco para el manejo de cadenas de caracteres. Esto encarece la facilidad con la que podrías trabajar en este desarrollador.
- ★ Falta de funciones: El lenguaje C no tiene liberación de memoria automática, lo cual significa que deberás hacerlo manualmente y estar atento en el momento que quieras hacer esta acción.
  Tampoco tiene soporte para la programación orientada a objetos y no permite multihilo, aquellas rutinas de serie que te podrían permitir aprovechar al máximo el procesador.
- ★ No recomendable para sitios web: A pesar de que hay varios programas web hechos con este lenguaje, el apartado y funciones que se elaboran con lenguaje C no tiene opciones inteligentes.
  Tiene una sobrecarga de operadores y solo cuenta con un soporte para la programación genérica. Si hacemos un balance entre las ventajas y desventajas del lenguaje C, el uso de web ineficiente es un punto en contra muy fuerte.

4) investigue y defina qué se entiende por:
   a) Compilador, Compiladores de C, gcc, etc

      - Un compilador es un tipo de traductor que transforma un programa entero de un lenguaje de programación a otro. Usualmente el lenguaje objetivo es código máquina, aunque también puede ser traducido a un código intermedio o a texto.

      - GCC es un compilador para GNU considerado un estándar para los Sistemas Operativos derivados de UNIX, de Código abierto Las siglas GCC significan GNU Compiler Collection (Colección de compiladores GNU). Como su nombre indica es una colección de compiladores y admite diversos lenguajes: C,C + +, Objective-C, Chill, Fortran, Ada y Java.

   b) Directivas del pre-procesador

      - Las directivas de preprocesador son la herramienta de reemplazo de texto, que se usa en el programa para reemplazar el texto por su valor.

Es diferente de variable. Siempre que se usa una variable en el programa, el compilador la entiende como un valor almacenado en alguna dirección de memoria.

Pero cuando se usa la directiva del preprocesador en el programa, el texto o el nombre de la directiva del procesador no se considera como algún valor o código en la memoria. El compilador los reemplaza por su valor real en el programa y luego compila el programa.

c) Variables, operadores, tipos de datos y estructuras en C

- Variables: En C las variables tienen un nombre que las identifica, y sirve para hacer referencia a ellas.

  También tienen un tipo, que es el tipo de datos que puede almacenar. El valor de las variables es, como su propio nombre indica, variable. Podemos alterar su valor en cualquier punto del programa.

- Operadores:
  - El operador de asignación es **=** .
  - Incremento **++** y decremento **--** unario. Los cuales son más eficientes que las respectivas asignaciones **+** y **-**.
  - El operador división **/** es para división entera y flotantes. Por lo tanto hay que tener cuidado.

- Tipos de Datos:
  - Tipo entero: representa números enteros con o sin signo, que estarán compuestos por los dígitos del 0 al 9.
    // int nombre_variable = valor;

  - Tipo real: Se emplean para representar números reales (con decimales).

    Para definir datos reales se antepone la palabra reservada *float* al identificador de la variable. ***// float numero;***

  - Tipo carácter: Este tipo de datos se emplea para representar un carácter perteneciente a un determinado código utilizado por el ordenador (normalmente el código ASCII).
    // Char identificador = 'valor';

- Tipo cadena de caracteres: una cadena de caracteres es un número de caracteres consecutivos (incluso ninguno) encerrado entre unos delimitadores determinados, que en el lenguaje C son las comillas dobles.
  // Char identificador[cantidad] = " mensaje ";

- Estructuras:
  La estructura es una colección de variables, es un tipo de dato definido por el usuario.
  Son también conocidas como Registros. Ayudan a organizar y manejar datos complicados en programas debido a que agrupan diferentes tipos de datos a los que se los trata como una sola unidad en lugar de ser vistas como unidades separadas.

d) Arreglos y punteros en C

- Un puntero es una variable que contiene la dirección de una variable.

  // Un puntero, en C, se declara de la siguiente forma:

  TIPO * nombre_puntero ;

  Los punteros se utilizan frecuentemente en C, debido a que por lo general llevan un código más compacto y eficiente de lo que se puede obtener en otras formas.

  Los punteros y los arreglos están relacionados íntimamente.

  - Existe una estrecha relación entre punteros y arreglos, tanto que pueden ser usados de forma casi indistinta.

  - Una variable de tipo arreglo(Array) puede considerarse como un puntero al tipo del arreglo.

  - Los punteros pueden ser utilizados en cualquier operación que involucre subíndices de arreglos.

- Los Arreglos son una colección de datos del mismo tipo.

// Un Arreglo en C se declara de la siguiente manera:

```
#define tamaño 100
…
int planilla[tamaño];
…
```

Sirve para manejar un número “n” de elementos en común, ya sea de tipos definidos por el Lenguaje,(“int” , ”float” , ”String”) así como aquellos definidos por el programador.

Un arreglo unidimensional es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales.

e) Funciones, parámetros, main, void, en C

- El código de un programa escrito o desarrollado en C se divide en funciones.
  Las funciones en un programa son entidades que dado un conjunto de datos, se les encarga realizar una tarea muy concreta y se espera hasta obtener el resultado.
  Una función en C es un fragmento de código que se puede llamar desde cualquier punto de un programa.

  Uno de los aspectos más importantes en el diseño de un programa es la división y agrupación de tareas.

- Los parámetros son variables locales a los que se les asigna un valor antes de comenzar la ejecución del cuerpo de una función.

  El mecanismo de paso de parámetros a las funciones es fundamental para comprender el comportamiento de los programas en C.

- Todo programa de C tiene una función principal que se debe llamar main.

La función main sirve como punto de partida para la ejecución del programa.
La cual normalmente controla la ejecución del programa dirigiendo las llamadas a otras funciones del programa.

- Función con tipo de retorno nulo (void). También se dice que es una función sin ningún tipo de retorno.

f) Control de flujo del programa, explique.

- Las proposiciones de control de flujo de un lenguaje especifican el orden en que se realiza el procesamiento..

Las estructuras de control de flujo, son instrucciones que nos permiten evaluar si se puede cumplir una condición o no, incluso nos puede ayudar a evaluarla **n** cantidad de veces.

g) Librerías, que son, qué objetivo tienen.

- Las **librerías** o **bibliotecas** (del inglés - **Library** -) son archivos que contienen porciones de código con un objetivo determinado, como por ejemplo: leer datos de entrada, imprimir en pantalla, tratamiento de números, entre otras funcionalidades. Estos son importados a la cabeza del código que estemos escribiendo, lo cual nos permitirá hacer uso de todas la funcionalidades que permita o que se hayan escrito en él. Los objetivos de las librerías o bibliotecas son, en un principio, la reutilización de código y la reducción del mismo, es decir, si necesitamos implementar “x” funcionalidad a nuestro programa una primera opción sería escribirla nosotros mismo teniendo que perder tiempo en solucionar los errores que se pueden presentar y además alargando el código, la otra opción (y objetivo de las librerías o bibliotecas) es solamente importar el archivo librería con las funcionalidades ya escritas y testeadas a nuestro código y hacer uso de ellas sin tener que reescribirlas, aumentando así las líneas de código.