LENGUAJES DE PROGRAMACIÓN

Programas: Secuencias ordenes/instrucciones que le indican a un dispositivo electrónico/informático, mediante lenguaje de programación, que hacer y en qué orden, sin ambigüedad y con finalidad concreta.

Lenguaje de programación: Lenguaje formal (símbolos, sintaxis y semántica) para crear programas informáticos.

Al **programar**, implementan algoritmos, métodos para la resolución de problemas. Dan de ser:

- Precisos: Sin ambigüedades o casos sin contemplar.
- Ordenados: instrucciones dispuestas en un orden a realizar.
- Finitos: Con un final.

Lenguaje maquina: 0 / 1.

Lenguaje de bajo nivel: Se representan 0 y 1 con lenguaje mnemónico (lenguaje ensamblador).

Lenguaje de alto nivel: Más próximos al humano (Java, PHP, Python).

Proceso de traducción de un programa en alto o bajo nivel a lenguaje máquina.

- Antes de ejecutar el programa (compilación). Un programa compilador traduce el código fuente, escrito por el programador, a un fichero ejecutable compuesto de 0 y 1. C y C++.
- Mientras se ejecuta el programa (interpretación). Un programa interprete traduce el código fuente, escrito por el programador, en tiempo real e instrucción a instrucción, leyéndola y convirtiéndola a 0 y 1 y ejecutándola… PHP y Python.

Topologías de red

Las principales son:

1. Bus
2. Anillo
3. Estrella

Conversión binario-decimal

Desarrollamos el numero como la suma de: Digito multiplicado por la base del sistema de numeración (2) elevada a la posición (comenzando por posición 0 de D→I).

Conversión decimal-binario

Dividimos entre la base sucesivamente hasta que el cociente sea menor que la misma. El numero binario equivalente será la secuencia del ultimo cociente seguida del último resto, del penúltimo resto…. Hasta el primer resto.

Sistema numeración hexadecimal

Sistema de numeración posicional que tiene como base el 16, 16 dígitos diferentes van del 0 al 9 seguidos de letra A (equivalente a 10_{10}) a la F (equivalente a 15_{10}).

Nos permite indicar grandes secuencias de numero binarios de manera equivalente y abreviada (indicar direcciones de memoria, operaciones de CPU…).

Cada digito en hexadecimal equivale a 4 dígitos (bits) en binario y, por tanto, con 2 dígitos en hexadecimal podemos expresar 8 dígitos binarios (1 byte).

COMANDOS LINUX

Comandos: Instrucción que se envía al SO, a través de una interfaz de línea de comandos.

Sintaxis: "$nombre_comando -parametros arg1 arg2… argN"

Comandos de ayuda → **"man":** muestra el manual/ayuda del comando que le pasemos como parámetro. Consultar funcionamiento del resto de comandos.

Comandos básicos

"ls": Muestra listado con ficheros y directorios de un directorio/carpeta determinada. Si no indicamos parámetros, muestra el contenido del directorio actual.

- Parámetro -l: Información mas detallada.
- Parámetro -a: Elementos ocultos del sistema.
- Parámetro *: Listado recursivo, mostrara el contenido de las carpetas contenidas en el directorio listado.

Caracteres comodines:

- **"*"** para sustituirlo por cualquier cadena de 0 o + caracteres. "ls h*" – Listar ficheros y directorios que empiecen por y tengan a continuación 0 o + caracteres.
- **"?"** para sustituirlo por cualquier carácter individual. "ls h?????" – Listar ficheros que empiecen por h y tengan exactamente 5 caracteres más.

"cd": Desplazarnos a un directorio.

- "cd nombre_directorio"

  *De forma relativa, para subir al directorio superior/padre, utilizar "cd .." (si utilizas "cd ." te quedas en la carpeta actual).

"mkdir": Crear un directorio vacío.

"rmdir": Borrar un directorio vacío.

"rm": Borrar un fichero.

- Borrar varios ficheros de un directorio → "*".
- Borrar directorios y subdirectorios forzando el borrado → "-rf".

Para crear un fichero:

- "touch nombre_fichero".
- "cat > nombre_fichero + "intro" + "texto_fichero" + "Ctrl+D".

"cp": Copiar un fichero → "cp fichero_origen fichero_destino".

*Copiar varios ficheros origen a la vez a un mismo directorio destino → "cp origen1 origen2 origen3 destino"

Copiar todos los ficheros de un directorio → "*".

Copiar los subdirectorios de manera recurrente → "-r".

"mv": Mover directorios y ficheros. Similar a "cp" pero el resultado elimina cada origen una vez copiado en destino.

CONTENIDO DE FICHEROS

Redirección

En vez de mostrar el contenido de un directorio en pantalla, quiero redireccionarlo a un fichero para guardar directamente esta salida en el fichero. "ls > fichero" → "ls > salida.txt".

La redirección ">" sobrescribe (destruye) el fichero destino. Si queremos añadir contenido al final de un fichero existente ">>" → "ls >> salida.txt".

"**echo":** Muestra por salida estándar, la cadena de texto que recibe como parámetro → "echo "texto"".

- Introduce un salto de línea después del texto. Si quiere evitarlos "echo -n "texto"".

"cat": Muestra el contenido de uno/varios ficheros en salida estándar → "cat texto.txt".

*Tanto "echo" como "cat" permiten el uso de redirección de la salida.

Visualizar el contenido de varios ficheros, indicando rutas separadas por un espacio → "cat salida.txt texto.txt".

"wc": Cuenta líneas ("-l"), palabras ("-w") o caracteres ("-c") de un fichero.

Tuberías → **"|":** Combinar varios comandos para ejecutarlos en un único mandato. La salida/resultado del 1er comando se envían a la entrada del 2do, y así sucesivamente.

"ps": Con "aux", nos muestra todos los procesos que hay ejecutándose en el sistema.

- Proceso: Elemento dinámico que representa una instancia de un programa en ejecución.

"kill": Finalizar un proceso en ejecución. Finalizar un proceso concreto → "kill -n".

"killall": Finalizar todos los programas de un tipo → "killall nombre_proceso".

"grep": Búsqueda de cadena de caracteres. Si generamos un fichero con una serie de nombres, podemos filtrar aquellas líneas que contengan una cadena concreta → "grep "cadena" ruta".

- Si queremos encontrar todas las cadenas que aparezcan en los ficheros de nuestro directorio actual → "grep "Jose" ./*".
- Si quisiéramos buscar dentro de los subdirectorios (búsqueda recursiva), con parámetro "-r".

"find": Buscar ficheros/carpetas en una ruta determinada.

"head": Acompañado de un numero natural como parámetro "-n", muestra las **"n primeras líneas"** de un fichero → "cat nombres.txt | head -3".

"tail": Similar a "head", nos muestra las **"n ultimas líneas"** de un fichero.

- "cat nombres.txt | tail -2".
- "cat nombres.txt | head -3 | tail -1".

AWK: Procesar patrones en líneas de texto.

- Acceder a datos con formato tabla.
- Acceder a datos separados por un delimitador.

Se considera un lenguaje por su potencialidad y versatilidad.

Leer ciertas columnas de una tabla → "awk '{print $numero_columna_1, …, $numero_columna_n}'".

Ejemplo → Ejecutamos "ls -l". Para que el comando **awk** muestre el valor de las **columnas 3 y 9**:

- "$ ls -l | awk '{print $3, $9}'.

Nos permite localizar coincidencias en el contenido de ficheros, dado un patrón de búsqueda. El patrón de búsqueda se denomina "**expresión regular**" → Denominación más genérica utilizada en otros ámbitos, y en detalle, es una secuencia de caracteres.

Sintaxis → "$ grep [opciones] [expresión regular] [archivos]".

- [opciones]: Indicar parámetros para personalizar uso de comandos.
- [expresión regular]: Definir el patrón búsqueda.
- [archivos]: Expresar los archivos (ruta) donde se realizará la búsqueda.

Buscar en el contenido de un fichero

Buscar cadena de texto en un fichero conocido → "$ grep "texto" fichero.txt" o "$ cat fichero.txt | grep "texto"".

*El comando devolverá las líneas del fichero que contengan alguna coincidencia con el patrón de búsqueda, la cadena "texto".

Opciones podemos indicar como parámetros:

- "-i": No tendrá en cuenta mayúsculas y minúsculas → "$ cat fichero.txt | grep -i "texto"".
- "-v": Devuelve resultado inverso, es decir, las líneas que no cumplan el patrón → "$ cat fichero.txt | grep -v "texto"".
- "-n": Numero de línea donde sucede la coincidencia → "$ cat fichero.txt | grep -n "texto"".
- "-W": Encuentra coincidencia solo si el patrón coincide con una palabra completa → "$ cat fichero.txt | grep -w "texto"".

*Se pueden combinar → "$ cat fichero.txt | grep -iw "texto"".

Expresión regular

Son una secuencia de caracteres que conforma un patrón de búsqueda. Permite definir una serie de aspecto con mayor nivel de detalle.

- "^": Indica que dicha cadena debe aparecer, al comienzo de una línea del fichero → "$ cat fichero.txt | grep "^texto"".
- "$": Igual que el anterior, pero al final del fichero → "$ cat fichero.txt | grep "texto$"".
- ".": Sustituye a cualquier carácter (similar al "?") → "$ cat fichero.txt | grep "t…o"".
- []: Lo que va dentro, se entiende como un conjunto de posibilidades que deben cumplir el resultado:
  - "$ cat fichero.txt | grep "[a-z]*texto"" → Busca una cadena formada por una letra desde la a hasta la z, seguida de la palabra "texto". Encontraría línea de fichero, si contiene → **Contexto o pretexto.**
  - "$ cat fichero.txt | grep "U[0-9]" → Busca una cadena formada por la letra "U" seguida de un numero de 0 a 9. Encontraría la línea del fichero si contiene → **"U1" o "U2".**
  - "$ cat fichero.txt | grep "[a-t]*texto"" → "*" indica que la posible letra se repite de 0 a n veces, en este caso acotando el conjunto desde la a hasta la t, y después va seguida la palabra "texto". Encontraría línea del fichero con → **Contexto, pretexto, texto…**
  - "$ cat fichero.txt | grep "U[0-1]\{1,2\}" → Busca una cadena formada por "U", seguida de 1 o 2 números acotados entre 0 y 1. Encontraría línea del fichero con → **"U0" o "U1" o "U10" o "U01".**

Buscar en múltiples ficheros → "grep [opciones] [expresión regular] [ruta_archivos]".

[ruta_archivos], puede ser:

- Todos los archivos de la carpeta actual (ruta relativa) → "./*".
- Archivos que empiezan por "lib", de la carpeta padre (ruta relativa) → "../lib*".
- Archivos texto de carpeta de usuario (ruta absoluta) → "/home/usuario/*txt".
- Cualquier otra ruta absoluta o relativa.

[opciones], puede ser:

- "-r": Búsqueda recursiva, para que busque también en los subdirectorios.
  - "$ grep -r "texto" ./*" → Busca cadena "texto" en todos los ficheros de la carpeta actual y en todos los subdirectorios. Nos dará la ruta de los ficheros en los que encuentre la cadena "texto" y, dentro de cada fichero, las líneas en las que ha localizado la coincidencia.

Ejemplo combinado

En base a "fichero.txt" que nos proporcionan y guarda datos personales.

Muestra el nombre y apellidos (columnas séptima y octava respectivamente).

Socios que contengan en sus datos el patrón siguiente:

- 5 dígitos numéricos entre 0 y 9, seguidos de una de las letras "L" o "H".

"$ cat fichero.txt | grep "[0-9]\{5}[]LH}" | awk '{print $7, $8}'"