Fundamentos de Sistemas Operativos

primera práctica

Explotación interactiva de "sistemas operativos de tipo UNIX" mediante la interacción con "intérpretes de comandos de tipo UNIX"

(shell interactivo)

Escuela Politécnica Superior de Alcoy
Curso 2021 – 2022

Primera práctica: shell interactivo

Objetivos

- Adquirir competencia en la explotación interactiva de “sistemas operativos de tipo UNIX” mediante la interacción con “intérpretes de comandos de tipo UNIX”.
- Asimilar el concepto “intérprete de comandos de tipo UNIX” y los principios de explotación UNIX “redirección de entrada/salida” y “combinación de comandos”.

Aspectos ejercitados

- Uso interactivo básico de “intérpretes de comandos de tipo UNIX”.
- Consulta interactiva de manuales (`man`).
- Consulta de información básica (`who`, `date` y `echo`).
- Nombrado de ficheros (`pwd`, `cd`, y uso de metacaracteres).
- Organización externa de ficheros (`ls`, `cp`, `mv`, `rm`, `mkdir`, `rmdir` y `find`).
- Exploración del contenido de ficheros (`cat`, `more`, `head`, `tail`, `grep`, `sort` y `wc`).
- Exploración de atributos de ficheros (`ls`).
- Exploración de atributos de procesos (`ps` y `top`).
- Archivado y compresión de ficheros y directorios (`tar` y `gzip`).
- Redirección de entrada/salida (los metacaracteres >, <, 2>, y >>).
- Combinación de comandos (el metacaracter |).

Contenido

Introducción

En esta introducción, se recuerdan conceptos básicos de sistemas operativos en general y de “sistemas operativos de tipo UNIX” en particular y se introducen, sin ánimo de sustituir a las páginas correspondientes del manual, diferentes comandos, típicamente disponibles en “sistemas operativos de tipo UNIX”, de aplicación generalizada.

1.1.- Los “intérpretes de comandos de tipo UNIX”.

En general, los intérpretes de comandos determinan la forma en la que se perciben y usan los sistemas informáticos. Además, en “sistemas operativos de tipo UNIX”, los “intérpretes de comandos de tipo UNIX” determinan la forma en que se edifica nueva funcionalidad, por lo que cobran, si cabe, mayor importancia.

En esta sección, se recuerda el concepto de intérprete de comandos, en general, la filosofía de explotación subyacente al diseño del sistema operativo UNIX, el concepto de utilidad UNIX, el concepto de “intérprete de comandos de tipo UNIX”, en particular, y el concepto de sesión de trabajo.

1.1.1.- Concepto de *intérprete de comandos*.

En general, hablamos de un “intérprete de comandos” cuando nos referimos a un servicio, de sistema operativo, típicamente identificado con un programa, cuyo propósito fundamental es interactuar con el usuario para poner a su disposición aquellos servicios, de sistema operativo, para los cuales se ha previsto que puedan ser accedidos a través del mismo. De manera que, en la medida en que el intérprete de comandos determina la forma en la que se solicitan los servicios, se convierte en un componente fundamental de la interfase usuario_humano – sistema_operativo, la cual se complementa con las interfases de los diferentes servicios, del sistema operativo, que, accedidos a través del intérprete de comandos, interactúan directamente con el usuario.

En un sistema operativo, el servicio “intérprete de comandos” no tiene porque ser único, siendo susceptible de ser implementado con diversas tecnologías de interacción con el usuario (textual, gráfica, oral…).

1.1.2.- Filosofía de explotación UNIX.

El diseño del sistema operativo UNIX responde a una forma específica de entender como explotar los recursos de un sistema informático y, en particular, a una forma de entender como reutilizar esfuerzo y edificar funcionalidad. Dicha forma de entender, dicha filosofía de explotación, se concreta en tres ideas fundamentales:

- Debería de ser posible el cambiar (redireccionar) el origen y/o el destino de la información que procesa un comando, en el momento de ordenarlo.
- Debería de ser posible el "combinar" el efecto de varios comandos.
- Debería de ser posible el utilizar un intérprete de comandos "a modo de lenguaje de programación".

Redirección de entrada/salida.

En general, el núcleo algorítmico que proporciona determinada funcionalidad es razonablemente independiente de la procedencia o destino de la información que procesa. Así, por ejemplo, el núcleo de un algoritmo que ordena una lista de palabras es independiente de la procedencia o destino de la misma (el terminal, un fichero...). Si fuera posible cambiar el origen y/o destino de la información que procesa un comando, en el momento de ordenarlo, el esfuerzo invertido, en el desarrollo del comando, podría reutilizarse en diferentes situaciones en las que los orígenes y/o destinos de la información procesada podrían ser diferentes.

Redireccionar la entrada es cambiar el origen de la información que procesa un comando, en el momento de ordenarlo.

Redireccionar la salida es cambiar el destino de la información que procesa un comando, en el momento de ordenarlo.

Combinación de comandos.

Una necesidad, podría ser, la de obtener un listado del contenido de un directorio ordenado alfabéticamente. Otra, la de obtener una relación de procesos ordenada por identificador de proceso. En un escenario de explotación convencional, ambos servicios requerirían de la implementación de algoritmos de ordenación. Alternativamente, el principio de explotación "combinación de comandos" (o "encadenado de comandos") sugiere la posibilidad de implementar la función de ordenación una sola vez. El listado del directorio ordenado se obtendría "combinando" el efecto de un comando que sería capaz de proporcionar el listado del directorio, sin ordenar, con el efecto del comando que es capaz de ordenar. Análogamente, la relación de procesos ordenada se obtendría "combinando" el efecto de un comando que sería capaz de proporcionar la relación de procesos, sin ordenar, con el efecto del comando que es capaz de ordenar. Además, la función de ordenación podría ser reutilizada en infinidad de otras ocasiones.

Lenguaje de programación intérprete de comandos.

La funcionalidad mínima atribuida a la parte del sistema operativo conocida como intérprete de comandos es la de interactuar con el usuario para poner a su disposición, en principio, de forma interactiva, un comando a la vez, aquellos servicios, de sistema operativo, para los cuales se ha previsto que puedan ser accedidos a través del mismo. A partir de aquí, la idea de proporcionar una "lista de comandos", por ejemplo en forma de contenido de un fichero de texto, "a modo de comando" cuyo efecto debería ser el de los diferentes que aparecen en la lista aplicados en secuencia, resulta inmediata. Con dicha funcionalidad extendida, los "comandos", que aparecerían en las listas, se asimilan a "instrucciones" de un lenguaje de programación y las "listas de comandos" a "programas", por lo que la notación entendida por el intérprete de comandos empieza a parecerse a un lenguaje de programación.

Para complementar y potenciar la idea de usar el intérprete de comandos a modo de lenguaje de programación, en los "intérpretes de comandos de tipo UNIX", se introduce, además, la posibilidad de utilizar variables y estructuras de control.

1.1.3.- Concepto de *utilidad UNIX*.

El conjunto de la filosofía de explotación UNIX, concretada en los principios de explotación "redirección de entrada/salida", "combinación de comandos" y "lenguaje de programación intérprete de comandos", sugiere el desarrollo de servicios, conocidos como *utilidades*, los cuales implementan una función básica, relativamente inespecífica, de aplicación generalizada, susceptible de ser matizada mediante "redirección de entrada/salida", combinada mediante "combinación de comandos", o integrada en un programa escrito en el lenguaje de programación "intérprete de comandos".

Dichas utilidades, deben presentar un diseño de entrada/salida independiente de dispositivo, la cual debe tener lugar, en principio, única y exclusivamente a través de la entrada/salida estándar, y deben estar implementadas de forma razonablemente eficiente y robusta.

1.1.4.- Concepto de "intérprete de comandos de tipo UNIX".

La función principal de un intérprete de comandos, también en UNIX (shell), es la de interactuar con el usuario para poner a su disposición los servicios básicos proporcionados por el sistema operativo. Sin embargo, dada la filosofía de explotación subyacente al diseño del sistema operativo UNIX, un "intérprete de comando de tipo UNIX" debe, además, permitir expresar y articular la redirección de la entrada/salida, permitir expresar y articular la combinación de comandos y ejecutar programas escritos en el lenguaje de programación que él mismo define.

Otras características típicas, de “intérpretes de comandos de tipo UNIX”, serían: la diferenciación entre comandos internos y externos, la posibilidad de ejecutar comandos en modo *background*, el uso de variables de entorno como forma de controlar el comportamiento de determinados comandos, el mantenimiento de históricos de comandos como forma de agilizar la interacción con el usuario, o el reconocimiento de metacaracteres (comodines o wildcards), usados para expresar patrones de nombres de ficheros.

La aproximación a la explotación de sistemas informáticos que representa el concepto de “intérprete de comandos de tipo UNIX” no es única. Sin embargo, resulta extraordinariamente potente, aunque requiere de cualificación. Contrasta con la otra forma de explotación típica, basada en intérpretes de comandos con soporte gráfico y dispositivos táctiles o ratón, más propia de interfases pensadas para usuarios no cualificados.

Una misma idea, la de un intérprete de comandos que es a su vez un lenguaje de programación, es susceptible de ser implementada, incluso para un mismo sistema, de diversas formas. Por lo que, típicamente, para un mismo “sistema operativo de tipo UNIX”, existen diferentes implementaciones de “intérpretes de comandos de tipo UNIX”. Es por ello que, con frecuencia, la expresión “intérprete de comandos de tipo UNIX” se utiliza en su forma plural.

Para “sistemas operativos de tipo UNIX”, el “intérprete de comandos de tipo UNIX” más básico es el *Bourne shell* (`sh`), que suele estar disponible en todas las instalaciones. Otros “intérpretes de comandos de tipo UNIX”, usuales, serían: el *Korn shell* (`ksh`), el shell con sintaxis parecida al lenguaje de programación C (`csh`) y el *Bourne-Again SHell* (`bash`), totalmente compatible con el `sh` al tiempo que incorpora características del `ksh` y `csh`.

1.1.5.- Concepto de *sesión de trabajo*.

Para poder interactuar con un intérprete de comandos, es necesario iniciar una sesión de trabajo, con el sistema operativo, sometiéndose a un proceso de identificación y autentificación de usuario (entrada en el sistema o *log in*) que, de completarse con éxito, da lugar a la puesta en servicio de un intérprete de comandos. La sesión de trabajo termina cuando, tras haber solicitado los servicios objeto de interés, se le comunica al intérprete de comandos que ya no hay intención de seguir interactuando con el sistema, desencadenándose el efecto de cierre de la sesión o salida del sistema (*log out*). De modo que, la sesión de trabajo viene delimitada por los eventos *log in* y *log out*.

Además, en cualquier momento, durante una sesión de trabajo, invocando el procedimiento de *log in*, es posible iniciar una nueva sesión de trabajo anidada, para la cual, llegado el momento, habrá que invocar el correspondiente *log out*.

1.2.- Los manuales en UNIX.

Al igual que ocurre con la mayoría de otros sistemas operativos, los manuales que documentan los servicios disponibles, en una instalación "de tipo UNIX," suelen estar instalados y accesibles para poder ser consultados en línea. Y, de hecho, las páginas de manual, accesibles en línea, deben ser consideradas la primera y principal documentación acerca de los servicios disponibles en una instalación.

1.2.1.- El comando `man`.

A la hora de aprender a usar un "intérprete de comandos de tipo UNIX", probablemente, lo primero que conviene conocer es la existencia del comando `man`, que sirve para consultar las páginas de manual. Y, seguramente, el primer comando que conviene ejecutar es el que informa acerca del uso del manual (que aparece en el ejemplo).

Ejemplo:

```
$  man man
```

1.3.- Consulta de información básica en UNIX.

Iniciada la sesión, puede tener interés averiguar con qué otros usuarios se comparte la máquina, si está esta en fecha y hora actualizadas, o cuál es el valor de las variables de entorno que determinan el comportamiento de diversos aspectos del desarrollo de la sesión.

1.3.1.- Los comandos `who`, `date` y `echo`.

`who`

El comando `who` proporciona la relación de usuarios que, en un momento determinado, comparten los recursos de un sistema, en modo *tiempo compartido*.

`date`

El comando `date` proporciona la fecha y hora establecidas como fecha y hora del sistema.

`echo`

El comando `echo` permite, entre otras cosas, consultar el valor de *variables de entorno*, como, por ejemplo, `HOME` o `PATH`, las cuales determinan el comportamiento de diversos aspectos del desarrollo de la sesión.

Ejemplo:

```
$ echo $PATH
```

1.4.- Nombrado de ficheros en UNIX.

Por lo general, para nombrar (o identificar) a una entidad en un *dominio de nombrado* (o *espacio de nombres*), suele ser suficiente con usar una cadena de caracteres alfanuméricos que respeta determinadas reglas sintácticas (típicamente, una longitud máxima y alguna restricción). Sin embargo, un sistema adecuado para nombrar ficheros es más complejo.

En esta sección: se recogen las reglas sintácticas para nombrar ficheros en UNIX, se presenta una solución al problema de nombrar ficheros, se recuerdan los conceptos *directorio de trabajo por defecto*, *directorio de inicio de sesión*, *pathname absoluto* y *pathname relativo*, y se introduce el uso de los comandos `pwd` y `cd`, y el uso de metacaracteres como forma de expresar patrones de concordancia con nombres de ficheros.

1.4.1.- Reglas sintácticas para nombrar ficheros en UNIX.

En “sistemas operativos de tipo UNIX”, la longitud máxima para un nombre de fichero varía entre 14 y 255. Además, conviene evitar la utilización de los caracteres **`; < > $ | * ?`**, ya que, en “intérpretes de comandos de tipo UNIX”, tienen significado específico. En general, pueden incluirse los caracteres alfabéticos, numéricos, el guión inferior (_) y el punto (`.`). Pero, conviene tener en cuenta que el punto, cuando se utiliza como primer carácter del nombre de un fichero, tiene el efecto de “ocultar” el fichero, a nivel de comando de generación de informes acerca del contenido de directorios (`ls`).

1.4.2.- Nombrado de un fichero en un directorio.

Una cadena de caracteres básica, sintácticamente correcta, permite nombrar a un fichero en el dominio de nombrado constituido por un directorio. Sin embargo, a menos que el *sistema de directorio*, del sistema de ficheros, *directorio único* resulte satisfactorio, en cuyo caso todos los nombres de ficheros colisionarían en un único dominio de nombrado, el problema de nombrar ficheros es más complejo.

1.4.3.- Nombrado de un fichero en un sistema de ficheros.

Para un sistema operativo multiusuario, como es UNIX, el *sistema de directorio*, del sistema de ficheros, *directorio único*, el cual obliga a nombrar a todos los ficheros del sistema con nombre diferente, resulta demasiado restrictivo. Se hace necesario soportar el mantenimiento de múltiples dominios de nombrado independientes (directorios). Una vez se contempla la posibilidad de la existencia de múltiples dominios de nombrado, el nombre, en su sentido básico (nombre base o *basename*), ya no es suficiente para referirse a un fichero. Resulta necesario especificar el directorio en el que el nombre identifica al fichero. En las versiones más generales de estas ideas (los sistemas de directorio *jerárquicos*), se permite el que los directorios sean, a su vez, nombrados en directorios. De modo que, llega a ser necesario especificar el directorio en el que se nombra al directorio en el que se nombra al directorio…en el que, en última instancia, se nombra al fichero.

1.4.4.- Concepto de *pathname absoluto*.

La expresión “pathname absoluto” (o “nombre ruta absoluto” o completo), en referencia a una forma de nombrar ficheros, especifica que, para nombrar a un fichero, hay que nombrar al directorio en el que se nombra al directorio en el que se nombra al directorio…en el que, en última instancia, se nombra al fichero, empezando desde el directorio raíz. Dicha expresión, referida a un fichero, se refiere al nombre del fichero, expresado en la forma descrita.

La mencionada forma de nombrar requiere de la adopción de un carácter separador, entre partes del nombre absoluto, el cual, “en el caso de sistemas operativos de tipo UNIX”, es la barra inclinada hacia delante / (*slash*).

En “sistemas operativos de tipo UNIX”, un nombre de fichero expresado como pathname absoluto siempre empieza por la barra inclinada hacia delante, la cual, en su primera aparición, más que servir de separador, representa al directorio raíz.

Ejemplo:

```
/home/users/usuario_1/practica_1
```

1.4.5.- Concepto de *directorio de trabajo por defecto*.

El pathname absoluto resuelve el problema de la identificación de ficheros en sistemas de directorio jerárquicos. Sin embargo, a poco que el sistema de ficheros alcance cierta complejidad (cierta profundidad), el nombrado de ficheros, mediante pathname absoluto, puede llegar a ser engorroso. Una forma de aligerar, la forma de nombrar, es establecer un directorio como “directorio de trabajo por defecto” (*working directory*). A modo de prefijo a añadir, a nombres ruta que no empiezan en la raíz, o entendiendo que la interpretación del nombre ruta se hará desde el directorio establecido como directorio de trabajo por defecto.

1.4.6.- Concepto de *pathname relativo*.

La expresión “pathname relativo” (o “nombre ruta relativo”), en referencia a una forma de nombrar ficheros, especifica que, para nombrar a un fichero, hay que nombrar al directorio en el que se nombra al directorio en el que se nombra al directorio…en el que, en última instancia, se nombra al fichero, empezando desde el directorio establecido como *directorio de trabajo por defecto*. Dicha expresión, referida a un fichero, se refiere al nombre del fichero, expresado en la forma descrita.

Es necesario el poder diferenciar cuándo un nombre ruta debe ser interpretado como absoluto o relativo. En “sistemas operativos de tipo UNIX”, la mencionada diferenciación se establece a partir del primer carácter del nombre ruta. Si el nombre ruta empieza por la barra inclinada hacia delante se interpreta como absoluto. En caso contrario se entiende que el nombre ruta es relativo (al *directorio de trabajo por defecto*).

1.4.7.- Concepto de *directorio de inicio de sesión*.

La expresión “directorio de inicio de sesión” (*home directory*) hace referencia al directorio que será fijado como *directorio de trabajo por defecto* cuando el usuario inicie una sesión.

1.4.8.- Los comandos `pwd` y `cd`.

`pwd`

El comando `pwd` permite conocer el *pathname* absoluto del directorio establecido como *directorio de trabajo por defecto*.

`cd`

El comando `cd` permite establecer a un directorio existente como nuevo *directorio de trabajo por defecto*.

Ejemplos:

```
$ pwd
/home/users/usuario_1
$ cd mi_directorio
$ pwd
/home/users/usuario_1/mi_directorio
```

1.4.9.- Uso de metacaracteres.

En ocasiones, interesa aplicar comandos a conjuntos de ficheros cuyo nombre concuerda con determinado patrón. Por ejemplo, “ficheros cuyo nombre empieza por la letra c”. En “intérpretes de comandos de tipo UNIX”, dichos patrones de concordancia se expresan haciendo uso de caracteres que tiene significado especial, llamados *metacaracteres* o *comodines*.

A efectos de expresar patrones de concordancia, los “intérpretes de comandos de tipo UNIX” atribuyen significado especial, entre otros, a los caracteres: `*`, `?`.

El caracter `*`

Concuerda con cualquier cadena de caracteres arbitraria, incluyendo la cadena vacía.

Ejemplo:

```
$ ls c*
```

El caracter ?

Concuerda con cualquier carácter individual.

Ejemplo:

```
$ ls practica_?.c
```

1.5.- Organización externa de ficheros en UNIX.

A diferencia de lo que ocurre con otros sistemas operativos, en sistemas operativos “de tipo UNIX”, nombre no es atributo de fichero. En UNIX, los nombres, por los que, a nivel externo, se identifica a los ficheros, y los ficheros mantienen identidad independiente. Es decir, los nombres se gestionan por un lado y los ficheros por otro. Es el servicio de directorio el que permite mantener asociaciones entre nombres y ficheros. Aunque, los ficheros no “están” en los directorios. Por ejemplo, es posible el que un mismo fichero pueda ser nombrado más de una vez, con nombre diferente, en un mismo directorio, e incluso con el mismo nombre, o con nombres diferentes, en directorios diferentes. Sin embargo, es imposible alcanzar situaciones en las que un fichero no se nombra en ningún directorio. La creación de un fichero implica la creación de una *entrada de directorio* en la que se le asocia un nombre.

La separación entre nombres y ficheros proporciona la máxima flexibilidad a la hora de facilitar el que los usuarios puedan controlar la forma en que, a nivel externo, perciben la organización de sus ficheros.

1.5.1.- Organización típica de un sistema de ficheros UNIX.

El sistema de ficheros de un “sistema operativo de tipo UNIX” suele presentar la siguiente organización:

El directorio raiz /

Es único en el sistema de ficheros que, en cada momento, es accesible, en un “sistema operativo de tipo UNIX”. Permite nombrar a ficheros y directorios.

/bin

Contiene los nombres de muchos de los ficheros ejecutables que implementan comandos externos UNIX.

/usr

Suele nombrar, entre otros, a los directorios de trabajo de los diferentes usuarios, aunque algunas instalaciones nombran dicho contenido en los directorios /users, /home o /home/users.

/etc

Suele nombrar a las implementaciones y archivos de configuración de los comandos empleados en la administración del sistema.

/dev

Contiene los nombres de los ficheros especiales que representan a dispositivos: terminales, líneas de comunicación, dispositivos de almacenamiento...

/proc

Contiene los nombres de los directorios en los que se nombra a los ficheros que contienen información acerca de los procesos que están activos en el sistema.

1.5.2.- Los comandos `ls`, `cp`, `mv`, `rm`, `mkdir`, `rmdir` y `find`.

Algunos de los comandos que permiten controlar la organización externa de ficheros, en UNIX, son: `ls`, `cp`, `mv`, `rm`, `mkdir`, `rmdir` o `find`.

`ls`

El comando `ls` genera informes, con diferente nivel de detalle, según opciones, acerca de atributos de ficheros y contenido de directorios.

`cp`

El comando `cp` copia ficheros, incluso varios en una sola invocación, dependiendo de opciones y modalidades de invocación.

`mv`

El comando `mv` permite cambiar el nombre por el que se conoce a un fichero, e incluso el directorio en el que se nombra (mover).

`rm`

El comando `rm` elimina entradas de directorio, nombres y, llegado el caso, ficheros y directorios, incluso de forma recursiva.

`mkdir`

El comando `mkdir` crea y nombra nuevos directorios.

`rmdir`

El comando `rmdir` elimina directorios vacíos.

find

El comando find permite localizar ficheros que cumplen determinadas propiedades.

Ejemplos:

```
$ ls
   ...
$ cp practica_* directorio_copia
$ mv directorio_copia/* otro_directorio
$ rm –r otro_directorio
$ mkdir nuevo_directorio_1 nuevo_directorio_2
$ rmdir directorio_copia
$ find / -user usuario_1
```

1.5.3.- Otros nombres para ficheros existentes (ln).

A diferencia de otros sistemas operativos, en sistemas operativos “de tipo UNIX,” es posible crear nuevos nombres para ficheros existentes, a los cuales nos podemos referir porque ya pueden ser nombrados, al menos, por un nombre. O dicho de otro modo, en “sistemas operativos de tipo UNIX”, un mismo fichero puede llegar a ser nombrado con diferentes nombres.

ln

El comando ln, en diferentes modalidades, permite establecer un nuevo nombre para un fichero existente.

1.6.- Exploración del contenido de ficheros en UNIX.

Siguiendo la filosofía de explotación UNIX, la exploración del contenido de ficheros en UNIX, se aborda haciendo uso de utilidades básicas, cuya funcionalidad puede ser combinada.

1.6.1.- Los comandos cat, more, head, tail, grep, sort y wc.

Algunos de los comandos que permiten explorar el contenido de ficheros, en UNIX, son: cat, more, head, tail, grep, sort o wc.

cat

El comando cat escribe, en la salida estándar, como una sola secuencia, el contenido leído, de la entrada estándar o de cada uno de los ficheros suministrados como parámetro.

more

El comando more permite formatear el contenido de una lista de ficheros para mostrarla en un terminal.

head

El comando head escribe, en la salida estándar, la parte inicial (cabecera) del contenido leído, de la entrada estándar o de cada uno de los ficheros suministrados como parámetro.

tail

El comando tail escribe, en la salida estándar, la parte final (cola) del contenido leído, de la entrada estándar o de cada uno de los ficheros suministrados como parámetro.

grep

El comando grep escribe, en la salida estándar, la parte del contenido leído, de la entrada estándar o de cada uno de los ficheros suministrados como parámetro, que concuerda con patrones especificados como parámetro.

sort

El comando sort escribe, en la salida estándar o en un fichero especificado como parámetro, el contenido leído, de la de la entrada estándar o de cada uno de los ficheros suministrados como parámetro, ordenado, a nivel de líneas, según el criterio especificado como parámetro.

wc

El comando wc escribe, en la salida estándar, la cantidad de líneas, palabras y/o caracteres correspondientes a la totalidad del contenido leído, de la entrada estándar o de los ficheros suministrados como parámetro.

Ejemplos:

```
$ cat prueba
   ...
$ head –n 10 prueba
   ...
$ grep palabra prueba
   ...
$ sort –r prueba
   ...
$ wc –w prueba
```

1.7.- Exploración de atributos de ficheros en UNIX.

La implementación de ficheros implica el mantenimiento de sus atributos: información acerca de su propietario, información relacionada con el control del acceso al mismo, su tamaño, fecha de creación, de última modificación...

Con frecuencia, se presenta la necesidad de consultar atributos de ficheros. En "sistemas operativos de tipo UNIX", la consulta de atributos de ficheros se lleva a cabo haciendo uso del comando `ls`.

1.7.1.- El comando `ls`.

`ls`

El comando `ls` genera informes, con diferente nivel de detalle, según opciones, acerca de atributos de ficheros y contenido de directorios.

En particular, en "sistemas operativos de tipo UNIX", el comando `ls` permite averiguar, por ejemplo: quién es el propietario y grupo propietario de un fichero, cuáles son sus *bits de permiso*, su tamaño o el número de i-nodo en el que se localiza su representación.

Ejemplo:

```
$ ls -l
```

1.8.- Exploración de atributos de procesos en UNIX.

La implementación de proceso implica el mantenimiento de sus atributos: identificador de proceso, usuario titular del proceso, atributos de planificación, de contabilidad de consumo de recursos…

Con frecuencia, se presenta la necesidad de consultar atributos de procesos. En "sistemas operativos de tipo UNIX", la consulta de atributos de procesos se lleva a cabo haciendo uso de los comandos `ps` y `top`.

1.8.1.- Los comandos `ps` y `top`.

`ps`

El comando `ps` genera informes, con diferente nivel de detalle, según opciones, acerca de atributos y listas de procesos.

`top`

El comando `top` permite monitorizar, con diferente nivel de detalle, según opciones, atributos y listas de procesos.

En particular, en “sistemas operativos de tipo UNIX”, el comando `ps` permite averiguar, por ejemplo: qué procesos están activos en el sistema, cuáles pertenecen a determinado usuario, cuál es la prioridad de determinado proceso, su estado o la cantidad de memoria de la que dispone.

Ejemplo:

```
$ ps –u usuario_1
```

1.9.- Archivado y compresión de ficheros en UNIX.

En ocasiones, para facilitar el almacenamiento y/o transporte, conviene archivar (empaquetar en un único fichero) y/o comprimir el contenido de ficheros y directorios. En sistemas operativos de tipo UNIX, ambas trasformaciones pueden conseguirse haciendo uso de los comandos *tar* y *gzip*.

1.9.1.- Los comandos `tar` y `gzip`.

`tar`

Fundamentalmente, el comando `tar` permite archivar el contenido de ficheros y directorios en un único fichero, aunque también permite comprimir el resultado.

`gzip`

Fundamentalmente, el comando `gzip` permite comprimir ficheros, aunque también permite acumular (concatenar) varios ficheros en uno.

Ejemplos:

```
$ tar -cf archive.tar foo bar
$ gzip archive.tar
```

1.10.- Redirección de entrada/salida en UNIX.

Un principio fundamental, de la filosofía de explotación UNIX, es el principio de explotación “redirección de entrada/salida”. Redireccionar la entrada (y/o salida), para un comando, es cambiar el origen (y/o destino) de la información que tiene que procesar, en el momento de ordenarlo. De modo que, un “intérprete de comandos de tipo UNIX” debe permitir expresar, y articular, la “redirección de entrada/salida”. En “intérpretes de comandos de tipo UNIX”, la redirección de entrada/salida se expresa haciendo uso de determinados metacaracteres.

1.10.1.- Los metacaracteres >, <, 2> y >>.

>

El metacaracter > permite expresar que el canal de salida estándar, del proceso en el que se ejecutará el comando que se especifica en la línea en la que aparece, debe ser redireccionado al destino que se especifica a la derecha de dicho metacaracter.

<

El metacaracter < permite expresar que el canal de entrada estándar, del proceso en el que se ejecutará el comando que se especifica en la línea en la que aparecen, debe ser redireccionado al origen que se especifica a la derecha de dicho metacaracter.

2>

El conjunto de símbolos 2> , usados a modo de metacaracter, permiten expresar que el canal de error estándar, del proceso en el que se ejecutará el comando que se especifica en la línea en la que aparecen, debe ser redireccionado al destino que se especifica a la derecha de dichos símbolos.

>>

El conjunto de símbolos >> , usados a modo de metacaracter, permiten expresar que el canal de salida estándar, del proceso en el que se ejecutará el comando que se especifica en la línea en la que aparecen, debe ser redireccionado al destino que se especifica a la derecha de dichos símbolos, con el matiz adicional, respecto a la redirección expresada con el metacaracter > simple, de que, en el caso de que el fichero destino de la redirección exista, el nuevo contenido debe añadirse al final del existente.

Ejemplos:

```
$ ls > fichero_listado
$ cat < fichero_listado
$ find . -user usuario_1 2> fichero_errores
$ cat fichero_listado >> fichero_errores
```

1.11.- Combinación de comandos en UNIX.

Otro principio fundamental, de la filosofía de explotación UNIX, es el principio de explotación “combinación de comandos” (o “encadenado de comandos”), cuya idea es edificar funcionalidad combinando (encadenando) el efecto de varios comandos.

Más concretamente, la “combinación” de dos comandos se articula “conectando” el canal de salida estándar del proceso en el que se ejecutará uno de los comandos, cuyo efecto se intenta combinar, con el canal de entrada estándar del proceso en el que se ejecutará el otro comando, objeto de la combinación. Dicha “conexión” se lleva a cabo a través de una abstracción para propósitos de comunicación entre procesos (*pipe*), la cual facilita el desacoplamiento de ritmo de evolución de los procesos, cuyo efecto se combina, asumiendo funciones de buffering. Obviamente, el mecanismo descrito se extiende fácilmente a la combinación (encadenado) de más de dos comandos.

De modo que, un “intérprete de comandos de tipo UNIX” debe permitir expresar, y articular, la “combinación de comandos”. En “intérpretes de comandos de tipo UNIX”, la “combinación de comandos” se expresa haciendo uso del metacaracter |.

1.11.1.- El metacarter | .

|

El metacaracter | permite expresar que los comandos especificados a ambos lados de dicho metacaracter deben ser ejecutados en procesos diferentes, interconectados por medio de una *pipe*.

Para provocar el efecto, atribuido al metacaracter |, el intérprete de comandos debe: crear los procesos en los que se ejecutarán los comandos especificados a ambos lados de dicho metacaracter, crear un pipe, redireccionar el canal de salida estándar, del proceso en el que se ejecutará el comando especificado a la izquierda del metacaracter, hacia la pipe, redireccionar el canal de entrada estándar, del proceso en el que se ejecutará el comando especificado a la derecha del metacaracter |, para leer desde la pipe, y desencadenar la ejecución de ambos comandos, los especificados a izquierda y derecha del metacaracter |, en los procesos correspondientes.

Ejemplos:

```
$ ps –u usuario_1 | wc -l
$ cat fichero_1 | grep resultado | sort –r
```

Ejercicios de autoevaluación

1 Escribir una orden que informe acerca del uso del intérprete de comandos *bash*.

2 Visualizar el *pathname absoluto* del directorio de inicio de sesión.

3 Escribir una orden que establezca como *directorio de trabajo por defecto* al *directorio de inicio de sesión*.

4 Estando establecido como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, escribir una orden para crear un directorio, llamarlo *autoevaluacion* y nombrarlo en el *directorio de inicio de sesión*.

5 Estando establecido como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, y suponiendo que hemos creado y nombrado en él un directorio llamado *autoevaluacion*, escribir una única orden capaz de crear un nuevo directorio, llamarlo *sub_autoevaluacion* y nombrarlo en el directorio *autoevaluacion*.

6 Estando establecido como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, y suponiendo que hemos creado y nombrado en él dos directorios, uno llamado *autoevaluación* y otro *copia_autoevaluacion*, escribir una única orden capaz de hacer una copia (y de nombrar en el directorio *copia_autoevaluacion*) de cada uno de los ficheros que, en el directorio *autoevaluacion*, tiene nombre que empieza por “a”.

7 Suponiendo que, en un sistema de tipo UNIX, como resultado de la ejecución del comando `ls –l`, obtenemos el siguiente informe:

```
-rwxr-xr-x   1  user_1   gusr     1024  Feb  5  2008  directorio2
-rw-r--r-x   1  user_1   gusr     1024  Feb  6  2008  directorio1
drwxr-xr-x   2  user_1   gusr     4096  Feb  6  2008  fichero1.c
-rw-r—r--    1  alum_1   alumnos  1148  Oct  3  2007  directorio4
-rw-r—r--    1  alum_2   alumnos  1020  Oct  1  2007  especial
brw-rw----   1  root     floppy  2, 72, Sep 18  2006  normal
```

Identificar, en dicho directorio, qué ficheros son: a) directorios, b) ficheros regulares c) ficheros especiales.

8 Escribir una orden para generar un fichero, llamado *usuarios*, con la relación de usuarios que están haciendo uso del sistema, de manera compartida, al mismo tiempo que nosotros.

9 Escribir la secuencia de órdenes necesaria para generar un fichero, llamado *usuarios*, cuyo contenido debe ser, en formato texto, la fecha en la que se ejecutó dicha secuencia de comandos y la relación de usuarios que estaban haciendo uso del sistema, de manera compartida, al mismo tiempo que nosotros.

10 Escribir una orden capaz de determinar el número de entradas del directorio */dev*, de la instalación “de tipo UNIX” a la que tienes acceso.

Soluciones a los ejercicios de autoevaluación

1 Escribir una orden que informe acerca del uso del intérprete de comandos *bash*.

```
$ man bash
```

2 Visualizar el *pathname absoluto* del directorio de inicio de sesión.

```
$ echo $HOME
```

3 Escribir una orden que establezca como *directorio de trabajo por defecto* al *directorio de inicio de sesión*.

```
$ cd $HOME
```

4 Estando establecido como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, escribir una orden para crear un directorio, llamarlo *autoevaluacion* y nombrarlo en el *directorio de inicio de sesión*.

```
$ mkdir autoevaluacion
```

5 Estando establecido como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, y suponiendo que hemos creado y nombrado en él un directorio llamado *autoevaluacion*, escribir una única orden capaz de crear un nuevo directorio, llamarlo *sub_autoevaluacion* y nombrarlo en el directorio *autoevaluacion*.

```
$ mkdir autoevaluación/sub_autoevaluacion
```

6 Estando establecido como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, y suponiendo que hemos creado y nombrado en él dos directorios, uno llamado *autoevaluación* y otro *copia_autoevaluacion*, escribir una única orden capaz de hacer una copia (y de nombrar en el directorio *copia_autoevaluacion*) de cada uno de los ficheros que, en el directorio *autoevaluacion*, tiene nombre que empieza por “a”.

```
$ cp autoevaluación/a* copia_autoevaluacion
```

7 Suponiendo que, en un sistema de tipo UNIX, como resultado de la ejecución del comando `ls –l`, obtenemos el siguiente informe:

```
-rwxr-xr-x   1  user_1   gusr     1024  Feb  5  2008  directorio2
-rw-r--r-x   1  user_1   gusr     1024  Feb  6  2008  directorio1
drwxr-xr-x   2  user_1   gusr     4096  Feb  6  2008  fichero1.c
-rw-r—r--    1  alum_1   alumnos  1148  Oct  3  2007  directorio4
-rw-r—r--    1  alum_2   alumnos  1020  Oct  1  2007  especial
brw-rw----   1  root     floppy  2, 72, Sep 18  2006  normal
```

Identificar, en dicho directorio, qué ficheros son: a) directorios, b) ficheros regulares c) ficheros especiales.

a) directorios: fichero1.c
b) ficheros regulares: directorio2, directorio1, directorio4 y especial
c) ficheros especiales: normal

8 Escribir una orden para generar un fichero, llamado *usuarios*, con la relación de usuarios que están haciendo uso del sistema, de manera compartida, al mismo tiempo que nosotros.

```
$ who > usuarios
```

9 Escribir la secuencia de órdenes necesaria para generar un fichero, llamado *usuarios*, cuyo contenido debe ser, en formato texto, la fecha en la que se ejecutó dicha secuencia de comandos y la relación de usuarios que estaban haciendo uso del sistema, de manera compartida, al mismo tiempo que nosotros.

```
$ date > usuarios
$ who >> usuarios
```

10 Escribir una orden capaz de determinar el número de entradas del directorio */dev*, de la instalación “de tipo UNIX” a la que tienes acceso.

```
$ ls /dev | wc –w
```

Ejercicios para evaluación

A continuación se propone una serie de ejercicios que el alumno puede realizar y, en su caso, entregar al profesor.

Algunos ejercicios a realizar dependen del número de DNI del alumno. Dicho número consta de 8 dígitos. Anotarlos, como se indica en el ejemplo, y adjuntarlos a la documentación presentada sobre las prácticas.

Ejemplo:

Digito	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
DNI	4	4	8	6	8	5	2	7

DNI alumno:

Digito	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
DNI								

Cada vez que, en un ejercicio, se indique *dígito_i* (siendo *i* un número entre 1 y 8), la expresión deberá ser sustituida por el valor consignado en la casilla, del DNI del alumno, correspondiente al dígito i. Así, según los datos mostrados en el ejemplo de DNI, si en un ejercicio se pide crear *dígito_3* directorios distintos, se deberá crear 5 directorios distintos (pues, en el DNI del ejemplo, tenemos el valor 5 correspondiendo al dígito 3). Si, al sustituir *dígito_i*, obtenemos 0 tomar 1 en su lugar.

Trabajo previo 1: Crear un directorio, llamado *primera_practica*, y crear, y nombrar en él, *digito_4* + 1 ficheros, con información variada. Cada uno de ellos debe contener un mínimo de *digito_2* líneas de texto.

Trabajo previo 2: Crear un directorio y llamarlo *copia*.

EJERCICIOS

1. Anotar la resolución de texto (número de filas y columnas), de la ventana con la que interactúas con el intérprete de comandos, y el texto que aparece en la *digito_4* línea escrita (no visualmente en blanco) de la página de manual correspondiente a la entrada *chmod* de la segunda sección del manual, de la instalación "de tipo UNIX" a la que tienes acceso.

2. Escribir una orden capaz de localizar el fichero ejecutable correspondiente a la implementación del comando externo *find*, ejecutarla y anotar una forma de nombrar dicho fichero, usando un *pathname* completo.

3. Fijado como *directorio de trabajo por defecto* el *directorio de inicio de sesión,* y entendiendo que no se cambia, escribir una única orden capaz de hacer la copia de seguridad de uno de los ficheros creados en el *trabajo previo 1*, nombrando la copia en el directorio *copia*, creado en el *trabajo previo 2*.

4. Fijado como *directorio de trabajo por defecto* el *directorio de inicio de sesión*, escribir una única orden capaz de hacer la copia de seguridad de todos y cada uno de los ficheros creados en el trabajo previo 1, nombrando, las copias, en el directorio *copia*, creado en el *trabajo previo 2*.

5. Escribir una orden que muestre las últimas *digito_2* líneas de texto de alguno de los ficheros creado en el *trabajo previo 1*. Ejecutar la orden y anotar el resultado.

6. Anotar el usuario propietario, el grupo de usuarios propietarios y los bits de permiso de la *digito_8* entrada del directorio */dev*, de la instalación "de tipo UNIX" a la que tienes acceso.

7. Escribir una orden para generar un informe, de procesos activos, en el que deben aparecer la totalidad de los procesos que están activos, en el sistema, en un momento dado. Ejecutar la orden y anotar la información proporcionada para el proceso que aparece *digito_3* líneas más arriba del que presenta el identificador de proceso (PID) más alto.

8. Escribir un orden para archivar, en un solo fichero, en formato comprimido, la mitad de los ficheros, a elegir, de los generados en el *trabajo previo 1*.

9. Haciendo uso de la posibilidad de redireccionar la entrada/salida, para un comando, y sin utilizar ningún editor de ficheros, diseñar una secuencia de comandos capaz de generar un fichero, llamado *mi_listado*, cuyo contenido debe ser, en formato texto: una fecha (la de el día de la ejecución de la secuencia de comandos), una hora (aproximadamente, la de la ejecución de dicha secuencia de comandos), la relación de usuarios que estaba usando el sistema cuando se ejecutó la referida secuencia de comandos, y el listado correspondiente al contenido del directorio *primera_practica*, creado en el *trabajo previo 1*. Además, ejecutar la secuencia de órdenes y anotar el contenido del fichero *mi_listado*.

10. Haciendo uso de la posibilidad de combinar el efecto de varios comandos en uno, escribir una orden que muestre las primeras *digito_4* + 1 entradas del contenido del directorio *primera_practica*, creado en el *trabajo previo 1*, ordenadas, alfabéticamente, de mayor a menor.