Prácticas de laboratorio de Telemática II

Práctica 1

Departamento de Ingeniería Telemática

(ENTEL)

Mónica Aguilar
Juanjo Alins
Oscar Esparza
Jose L. Muñóz
Marcos Postigo
Antoni X. Valverde

La composición de este manual de prácticas ha sido realizada con el programa LaTeX 2ε. Agradecer a DONALD KNUTH la idea y el programa TeX y a LESLIE LAMPORT sus magníficas macros que han derivado en LaTeX.

Barcelona a 28 de Septiembre de 2002

• Revision 1.6 unix.lyx

Índice de las prácticas

Índice de figuras

PRÁCTICA 1

Entorno UNIX

1.1. Objetivo de la Práctica

1.1.1. Entorno de Trabajo: Topología de la red

Las prácticas se realizarán mediante 10 estaciones de trabajo conectadas a través de diferentes topologías (según los requisitos de las sucesivas prácticas) basadas en redes de área local tipo Ethernet.

La red Ethernet mediante la cual conectamos todos los ordenadores, tiene una topología en bus, utiliza un protocolo de acceso al medio CSMA/CD y nos permite transmitir datos a una velocidad de 10 Mbits por segundo. Este tipo de red está especificado en la norma IEEE 802.3. Algunas características de esta norma son que se ha de utilizar una codificación Manchester y la distancia máxima entre dos estaciones de trabajo no puede ser superior a 2,5 Km.

La limitación en la longitud de la Ethernet imposibilita su uso fuera de las LANs. De todas formas, siempre es posible enlazar varios segmentos de Ethernet entre sí utilizando repetidores (*hubs* y *switches*), puentes (*bridges*) y encaminadores (*routers*). Los repetidores simplemente copian las señales entre dos o más segmentos, de manera que todos los segmentos juntos actúan como si fueran una única Ethernet. Los *brigdes* y *routers* son más sofisticados, analizan los datos de entrada y los reenvían solo si el nudo receptor está en la Ethernet local.

Ethernet funciona como un sistema de bus, donde un nodo puede enviar paquetes de hasta 1500 bytes a otro nodo de la misma Ethernet. A cada nodo se le asigna una dirección de seis bytes grabada en el *firmware* (memoria fija) de su tarjeta Ethernet. Estas direcciones de red (también llamadas *mac address*) se especifican normalmente como una secuencia de números hexadecimales de dos dígitos separados por dos puntos, como por ejemplo aa:bb:cc:dd:ee:ff.

Las Ethernets se pueden clasificar en tres tipos: gruesas, finas y de par trenzado. Las dos primeras pueden utilizar cable coaxial, difiriendo en el grosor y en el modo de conectar el cable a los nodos. La Ethernet fina hace servir conectores "BNC" con forma de T que se pinchan al cable y que se conectan por detrás del ordenador. La Ethernet gruesa requiere que se haga un agujero al cable y que se conecte un transceptor utilizando un conector especial denominado “conector vampiro”. Después se podrán conectar uno o más nodos al transceptor. Los cables finos y gruesos pueden conseguir distancias de 200 y 500 metros respectivamente. Por eso también se los conoce también como 10base-2 y 10base-5. La palabra "base" hace referencia a la modulación en banda base y significa que los datos pasan directamente al cable sin pasar por un modem. El número de delante hace referencia a la velocidad en Mbps, mientras que el número del final indica la máxima longitud que se le puede dar al cable en centenares de metros. El par trenzado hace servir un cable de dos hilos de cobre. A esta Ethernet se la conoce también como 10base-T, donde "T" indica de par trenzado. Los pares trenzados con velocidad de 100 Mbps se conocen como 100base-T.

Los alumnos disponen de una estación de trabajo por pareja. Las estaciones de trabajo son de arquitectura 386. Su configuración y prestaciones son las siguientes:

- CPU Intel Pentium III (Coppermine) 900 Mhz
- 256 MB de memoria RAM
- 1 Floppy disk

- 1 CD ROM
- Disco duro de 40 GB
- Tarjeta gráfica SVGA de 1024*768
- Monitor color de 17”
- 2 tarjetas de red 10/100 Mbps

La Figura 1.1 muestra la estructura física del laboratorio. Como se observa, cada par de máquinas comparte un *hub* y tiene dos conexiones de red en la pared (las otras dos conexiones están inutilizadas y no se utilizarán en las prácticas). Las conexiones de la pared enlazan directamente con las conexiones de la zona de profesores, donde se realizará el conexionado necesario para cada práctica.

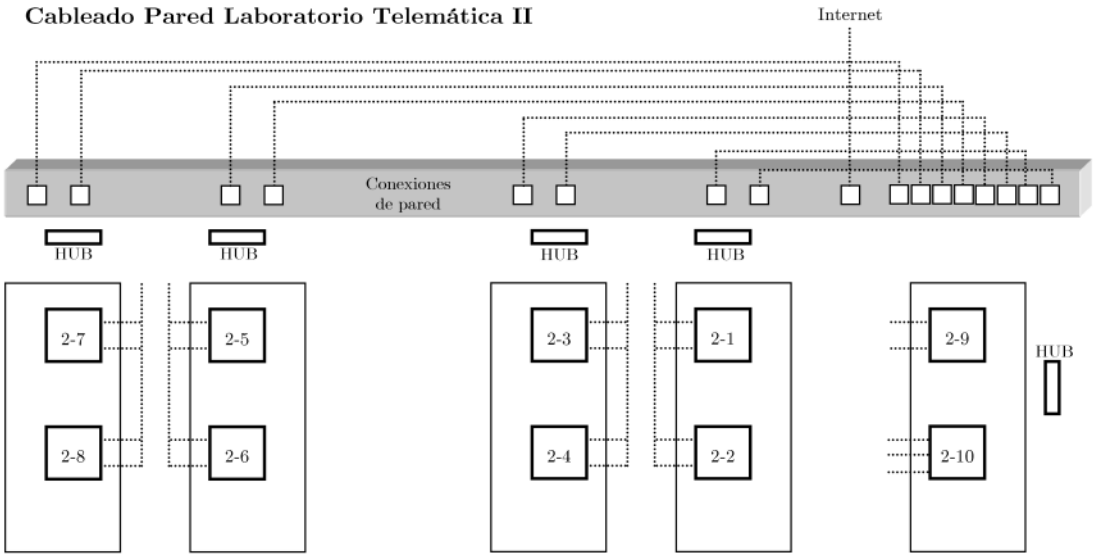

Figura 1.1: Cableado del laboratorio.

1.2. El Sistema Operativo UNIX

1.2.1. Introducción

UNIX fue diseñado para trabajar en red y ya nació con los protocolos TCP/IP en su configuración, cosa que hace de UNIX un sistema ideal para trabajar con redes de ordenadores. No obstante, UNIX actualmente es una filosofía más que un sistema operativo propio. De UNIX han derivado muchas implementaciones (es decir, diferentes sistemas operativos de estilo UNIX) de diversas empresas del sector informático. Algunas de ellas están apoyadas y/o desarrolladas por empresas de *hardware* (Solaris de Sun Microsystems, AIX de IBM, SCO de Unisys, IRIX de SGI, etc.) y otras distribuciones de UNIX no son comerciales, como es LINUX. Distribuciones de Linux que podemos encontrar son SuSE (la que se hará servir en estas prácticas), RedHat, Caldera, Debian, Mandrake, etc. Nuestras máquinas funcionan con el sistema operativo Linux, (en concreto con la distribución Debian de Linux).

1.2.2. Características generales y funcionamiento básico

UNIX es un sistema diseñado para funcionar en red. Además, UNIX permite gestionar diferentes usuarios a la vez (multiusuario) y realizar diversas tareas a la vez (multitarea). Por ello necesita de un proceso para controlar la entrada de cada uno de los usuarios registrados en el sistema, desde cualquier punto de la red. Este proceso es el programa *login*, con el siguiente aspecto para el usuario que entra al sistema:

```
telem2-x login:
```

El primer campo es el nombre de la máquina, en este caso telem2-x (x entre 1 y 8), y el segundo campo nos solicita el nombre del usuario que quiere entrar al sistema. A cada pareja de alumnos se le dará un nombre de usuario del tipo LT2xx, siendo xx dos números proporcionados por el profesor en clase.

Una vez hemos entrado el nombre de usuario, nos pedirá la contraseña asignada por el profesor (*password*). Si la tecleamos correctamente podremos entrar al sistema, pero si nos equivocamos la rechazará y tendremos que volver a empezar. Cuando escribimos la contraseña no aparece en pantalla por seguridad. La primera vez que entramos en el sistema, deberemos modificar el *password* asignado por uno propio del grupo (fácil de recordar por ambos, pero lo suficientemente complicado para que no sea fácil de averiguar) mediante el comando *passwd*.

```
password: ********
```

Además, UNIX es un sistema multitarea capaz de gestionar procesos de forma concurrente. Para la gestión de procesos dispone de un programa principal que gestiona todo el funcionamiento del sistema, se trata del núcleo o *kernel* del sistema operativo.

El *kernel* sabe cuántos usuarios hay conectados en cada momento, así como qué procesos están ejecutando cada uno de ellos. El *kernel* se encarga de dar prioridades a los procesos, de controlar los dispositivos básicos de entrada/salida, de gestionar la memoria, de repartir el tiempo de la CPU entre los diferentes procesos, etc. Para este último propósito, el *kernel* dispone de un *scheduler* o planificador que gestiona el intercambio de los procesos en el sistema.

1.2.3. Sistema multiusuario

UNIX es un sistema multiusuario, lo que quiere decir que en cada momento puede haber más de un propietario de las tareas y de la información que se almacena en el sistema. Cada usuario tiene la sensación de tener la máquina para el solo, pero lo que sucede en realidad es que hay una división del tiempo de utilización del procesador (*time sharing*) para cada usuario y un fuerte sistema de protección entre los datos y tareas que pertenecen a cada usuario. Dado que es posible que hayan procesos de varios usuarios funcionando en una misma máquina, es necesario **NO APAGAR LAS MÁQUINAS DEL LABORATORIO**. De igual forma, **EN CASO DE ENCONTRAR UNA MÁQUINA APAGADA SE DEBERÁ AVISAR AL PROFESOR**.

Un sistema UNIX permite el acceso simultáneo a centenares de usuarios. Se distinguen dos tipos de usuario: el superusuario o *root* y los usuarios normales. El *root* es el encargado de la administración del sistema, tiene derechos para realizar cualquier tipo de tarea y para acceder a cualquier fichero. Los usuarios normales tienen ciertas restricciones en la utilización del sistema y en el acceso a ciertos directorios.

Dado que UNIX soporta múltiples usuarios y servicios, es necesario disponer de seguridad. El sistema está concebido de manera que un usuario convencional muy difícilmente pueda comprometer la seguridad de los otros usuarios y del sistema. El mecanismo básico de seguridad se basa en la asignación de palabras secretas o contraseñas a los diferentes usuarios, la existencia de los permisos y restricciones, y el control mediante el almacenamiento de la actividad que se produce en el sistema (los denominados *logs*). Dado que los sistemas están conectados a una red, hace que puedan ser accesibles incluso a veces desde cualquier parte del mundo, incrementando aún más los requerimientos de seguridad.

1.2.4. Sistema multitarea

Como sistema multitarea, UNIX permite a cada usuario ejecutar varias tareas simultáneamente. Por ejemplo, un usuario puede estar editando un texto, mientras ejecuta un largo programa de cálculo numérico y lanza a la vez una tarea de copia de un fichero, un navegador para consultar páginas *web*, etc.

Cualquier actividad en un sistema UNIX se conoce como proceso. Los procesos pueden ejecutarse en primer nivel (*foreground*) de forma interactiva, o en segundo nivel (*background*) sin utilizar la entrada/salida del terminal. Esto es útil para lanzar uno o varios procesos largos y podernos ir cerrando nuestra sesión.

Cuando un proceso activa a otro proceso, al primero se le llama proceso padre (*parent process*) y al segundo proceso hijo (*child process*).

Los procesos pueden presentar diversos estados: en marcha, parado, dormido (esperando algún suceso o resultado), *zombie* o *defunct* (cuando ya ha acabado pero no ha estado eliminado completamente del sistema, normalmente porque el proceso padre no lo ha atendido adecuadamente).

Los procesos pueden tener diferentes privilegios (derechos para hacer cosas) generalmente heredados de los derechos de su propietario, así como diferentes prioridades (tiempo de CPU de que disponen).

Un tipo de procesos especialmente importantes son los llamados *daemons* (o demonios). Normalmente funcionan en segundo nivel (sin terminal asociado) durante períodos de tiempo extensos (a menudo durante todo el tiempo que el sistema está en funcionamiento) y frecuentemente en estado dormido a la espera de algún suceso. Se utilizan en tareas básicas del sistema, como almacenaje de actividades y de errores (*klogd*, *syslogd*), gestión de colas de impresión (*lpd*), arrancar tareas planificadas a lo largo del tiempo (*atd*, *crond*), y para atender los servicios que requieren las comunicaciones (*inetd*, *telnetd*, *ftpd*, *smtpd*, *httpd*, ...).

1.2.5. Herramientas básicas de UNIX

Además del *kernel*, el sistema operativo consta también de muchas aplicaciones que son las que nos permiten trabajar en un entorno UNIX. Estas aplicaciones se invocan mediante comandos UNIX. Algunos comandos básicos que nos serán muy útiles, son (las opciones entre paréntesis son opcionales) los siguientes:

Comandos Básicos

`login`: sale de la sesión en la que estuviera y hace aparecer de nuevo el *prompt login*:

`ps`: muestra los procesos que corren en la máquina.

`pstree`: muestra un árbol de los procesos que corren en la máquina.

`top`: muestra el uso que se está haciendo tanto de la memoria como del procesador.

`su` *`user`*: permite cambiar de usuario. El nuevo usuario será *`user`*.

`whoami`: nos da el nombre del usuario dueño de la *`shell`* donde estamos ejecutando este comando.

`who`: nos da el nombre de todos los usuarios que actualmente tienen abierta una sesión en la máquina.

`passwd`: permite cambiar la contraseña del usuario, en la máquina.

`cat` *`file`*: nos presenta por pantalla el fichero *`file`*.

`more` *`file`*: nos presenta por pantalla el fichero *`file`* paginándolo por pantallas.

`less` *`file`*: similar a **`more`** pero permite moverse hacia atras con los cursores.

`cp` *`file_source`* `(-r)` *`file_destination`*: copia el fichero *`file_source`* con el nombre *`file_destination`*. Si el fichero *`file_source`* es un directorio, entonces con la opción `-r` especificamos que se copien también los ficheros que contiene dicho directorio, de forma recursiva.

`mv` *`file_source file_destination`*: mueve el fichero o directorio *`file_source`* a *`file_destination`*. Puede utilizarse para cambiar el nombre del fichero, que pasará a llamarse *`file_destination`*.

`rm (-r) file`: borra el fichero *`file`*. Con la opción `-r` también borramos los ficheros contenidos en *`file`* en el caso de que sea un directorio.

`ls (-l -a)`: lista el directorio actual. Para listar el contenido de un directorio, escriba `ls` *`directorio`*. La opción `-l` da un listado con más información de cada fichero (propietario, grupo, tamaño, fecha última modificación, etc.). La opción `-a` enseña los ficheros ocultos, que son todos los que comienzan con un punto, por ejemplo el fichero *.bash_rc*. Cuando estamos en un directorio y listamos su contenido mediante el comando `ls` con la opción `-l`, obtenemos un listado de filas como ésta:

```
drwxr-xr--  T2usr9  LT2   512  Sep 5 2002  directorio_prueba
```

Los 10 primeros caracteres se refieren a los permisos de ese fichero o directorio. Después aparece el propietario, el grupo al que pertenece, el tamaño en *bytes*, la fecha y el nombre.

`chmod mode file`: cambia los permisos del fichero *`file`* a los nuevos dados por *`mode`* (un número de tres cifras) que se ha obtenido de la suma de los nuevos permisos que damos. Siguiendo con el ejemplo anterior, los 10 primeros caracteres indican los permisos para ese fichero (si el primer carácter es un guión) o directorio (si por el contrario aparece la letra *`d`*). Las tres primeras letras *`rwx`* indican que el amo tiene permisos de lectura, escritura y ejecución. Los tres caracteres siguientes se refieren a los usuarios del mismo grupo que el amo, que en este caso tienen permisos de lectura y ejecución (*`r-x`*). Y los tres últimos caracteres indican que el resto de usuarios tienen sólo permiso de lectura (*`r--`*). Los permisos tienen un peso numérico: el de lectura (*read*) pesa 4, el de escritura (*write*) pesa 2 y el de ejecución (*execute*) pesa 1. Así, para especificar o modificar los permisos de un fichero o directorio, utilizaremos el comando `chmod` con el valor de *`mode`* igual a un número de tres cifras. La primera cifra es la suma de pesos para el amo, la segunda cifra es la suma de pesos para el grupo, y la tercera cifra es la suma de pesos para el resto del mundo.

Estos permisos son los de la tabla siguiente:

```
400 leer por el propietario
200 escribir por el propietario
100 ejecutar por el propietario
040 leer por el grupo
020 escribir por el grupo
010 ejecutar por el grupo
004 leer por el resto del mundo
002 escribir por el resto del mundo
001 ejecutar por el resto del mundo
```

de este modo si queremos que un fichero lo puedan leer todos los usuarios, pero que sólo lo pueda escribir el amo, escribiremos:

```
chmod 644 nombre_fichero
```

`pwd`: nos indica el directorio actual donde estamos. El nombre proviene de *print work directory*.

`cd path`: cambiamos el directorio actual a *`path`*. Si *`path`* es `".."` retrocederemos un directorio (directorio padre). Si omitimos *`path`* entonces nos situamos automáticamente en nuestro directorio por defecto (el directorio *`home`*). Si el directorio tiene un nombre muy largo, es muy útil escribir las primeras letras (hasta que no haya otro directorio que empiece igual) y usar el tabulador. Automáticamente se acabará de escribir el nombre.

`mkdir directory`: creamos un nuevo directorio con el nombre *directory*.

`rmdir directory`: borramos el directorio *`directory`*.

`man command`: proporciona ayuda sobre el comando *`command`*.

`echo string`: escribe la cadena de caracteres *`string`* en la pantalla.

`*`: substituye a cualquier número de caracteres. Por ejemplo, `ls f*`, listará a todos los ficheros que empiecen por la letra "`f`".

?: substituye a un único carácter. Por ejemplo, `ls f???`, listará a todos los ficheros que tengan un nombre de cuatro letras y empiecen por la letra "f".

Comandos para comprimir

Además de estos comandos, UNIX tiene unos símbolos que también realizan ciertas tareas, que son:

`tar -cvf fichero.tar directorio/`: agrupa todos los ficheros del directorio en un único fichero. Se utiliza para transmitir ficheros por Internet vía el protocolo de transferencia de fichero *(*`ftp`), normalmente junto a una previa compresión de ficheros con el comando `gzip`.

`tar -xvf fichero.tar`: extrae todos los ficheros del archivo *`fichero.tar`* y restablece la estructura original de directorio.

`tar -zxf nombre.tar.gz`: Como se ha comentado, es típico usar *`tar`* y *`gzip`* de forma consecutiva para compactar y comprimir ficheros. El comando *`tar`* tiene la opción *`z`* que hace ambas tareas. Aquí se usa para extraer los ficheros originales.

`gzip fichero`: comprime el fichero (que es borrado y substituido por el fichero comprimido) utilizando así menos espacio. El fichero comprimido se llama *`fichero.gz`*.

`gzip -d fichero.gz`: descomprime el *`fichero.gz`* (que es borrado y substituido por el fichero descomprimido).

Comandos para redireccionar

En UNIX, un comando tiene asignado una serie de dispositivos de salida y de entrada por defecto. Por ejemplo, cuando se ejecuta el comando `echo string`, el terminal muestra el resultado del comando en la pantalla, porque la salida estándar asociada a ese comando es la pantalla. En general, un comando UNIX tiene asociado una salida estándar, una salida de errores y una entrada estándar que se corresponden con el terminal para la salida estándar y la salida de errores, y el teclado para la entrada estándar. En UNIX se pueden redireccionar las entradas y salidas estándar de los comandos a ficheros o a otros comandos, de la siguiente manera:

`comando > fichero_de_salida`: redirecciona la salida del comando al fichero *fichero_de_salida*. Ejemplo: **`cat fichero >fichero2`** escribe el contenido de `fichero` en `fichero2`.

`comando < fichero_de_entrada`: redirecciona la entrada del programa al fichero *`fichero_de_entrada`*. Ejemplo: **`more <`** ***`fichero`*** visualiza fichero por pantallas; es equivalente a **`more`** ***`fichero`***.

`comando1 | comando2`: redirecciona la salida del `comando1` a la entrada del `comando2`. Ejemplo: **`cat`** ***`fichero`*** **`| more`** visualiza fichero por pantallas.

`comando >>` ***`fichero_de_salida`*****:** añade la salida del comando, al fichero *fichero_de_salida*. Ejemplo: **`cat`** ***`fichero3`*** **`>`** ***`fichero2`*** añade a *`fichero2`* el contenido de *`fichero3`*.

Comandos para exportar DISPLAY

UNIX permite seleccionar qué monitor gráfico (*display*) utiliza un proceso para mostrar los datos. Esto permite seleccionar un monitor en otra máquina distinta de la local, lo cual es útil cuando se realiza una conexión a una máquina remota y se quieren ver los resultados gráficos en la máquina local.

`xhost +`: habilita a cualquier máquina remota para utilizar la máquina local como monitor destino.

`export DISPLAY=direccion_maquina_origen:0.0` : cambia la variable de entorno `DISPLAY` (monitor destino) a la que se encuentra en **`direccion_maquina_origen`**.

1.3. Ejercicios

Ejercicio 1

Procesos

1. Vamos a lanzar procesos en *background*. Abra tres Command-Tool (ventanas de comandos). En una ejecutar `xeyes -geometry 400x400 -center red`, y en la otra `xclock &` (el símbolo `&` indica que se ejecute el proceso en *background*). Observe la diferencia en el *prompt* de ambas ventanas. Utilice la tercera ventana para ver qué procesos usted está ejecutando. Para ello utilice el comando `ps -aux`. Averigüe para qué sirven los parámetros `a u x`, para ello consulte la entrada al manual del comando `ps` (`man ps`) o bien pida la ayuda del comando (`ps --help`)
2. Con el comando `jobs` observe qué tareas usted ha abierto. Anote el número de tarea JID (Job IDentifier) y el nombre que tiene cada una. Puede consultar la ayuda de este comando mediante `man jobs`.
3. Con el comando `ps -aux` observe qué procesos corren en su máquina. ¿Qué comando ha de utilizar para ver sólo sus propios procesos?
4. Ejecute el comando `pstree -lupa`. Indique el resultado del comando y explique qué hace cada una las opciones (`-lupa`).
5. Haga que también el programa `xeyes` pase a *background*, para así recuperar el control del *prompt*. Para ello deberá detener el proceso (CTRL-Z) y ponerlo en *background* (`bg`). ¿Cómo debería haber lanzado el proceso `xeyes` desde el inicio para estar en el estado en *background*? ¿Cuál es el comando adecuado para ello? Hágalo para comprobarlo.
6. Ahora ponga dicho proceso en *foreground* (`fg`).
7. Elimine (comando `kill`) ambas tareas. Para el proceso `xeyes` utilice el PID (Process IDentifier) mediante el comando `kill -9 pid`. La opción `-9` fuerza la eliminación del proceso y de sus hijos. Para el poceso `xclock` utilice el JID mediante el comando `kill %jid`. Comente ambos resultados.
8. Ejecute el comando `ps -aux | grep su_nombre_usuario`. ¿Qué hace dicho comando? ¿Qué hace el comando `grep`? ¿Qué utilidad tiene?
9. Ahora ejecute el comando `ps -aux | grep -v su_nombre_usuario`. ¿Qué hace dicho comando? ¿Qué hace el parámetro `-v`? ¿Qué utilidad tiene?

Ejercicio 2

Redireccionar procesos

1. Vaya escribiendo los comandos necesarios para ir realizando las siguientes tareas. Sitúese en su directorio de trabajo. Desde ahí, haga un listado del contenido del directorio `/home/labt2`, con las opciones necesarias para obtener más información de cada fichero y para listar recursivamente todos los subdirectorios encontrados. Utilice la entrada al manual para consultar cuáles son las opciones necesarias. ¿Cuál ha sido el comando utilizado?
2. Ahora redireccione el resultado que ofrece el comando anterior escribiéndolo en un fichero de salida llamado `arbol.txt`. Visualice desde consola el contenido de dicho fichero. ¿Cuáles han sido los comandos utilizados?
3. Utilice el comando adecuado para escribir en la pantalla la frase "*Esta es la estructura de directorios de mi zona de trabajo.*". ¿Cuál ha sido el comando utilizado?
4. Ahora redireccione la salida del comando anterior de manera que esa frase se añada al final del fichero `arbol.txt`. ¿Cuál ha sido el comando utilizado?

Ejercicio 3

Permisos

1. Vaya escribiendo los comandos necesarios para ir realizando las siguientes tareas. Sitúese en su directorio de trabajo. Desde ahí, copie el fichero `arbol.txt` en el directorio `/usr`. ¿Qué ha pasado y por qué? ¿Cuál sería la solución necesaria para poder realizar esa copia?

2. Sitúese en su directorio de trabajo y cree una estructura de directorios tal como se muestra en la figura 1.2. Observe los permisos de cada uno de los directorios que usted ha creado. Elimine su propio permiso de escritura del directorio `sin_permiso`. Intente los comandos `cp arbol.txt con_permiso` y `cp arbol.txt sin_permiso`. Comente qué ha sucedido en cada caso y por qué.

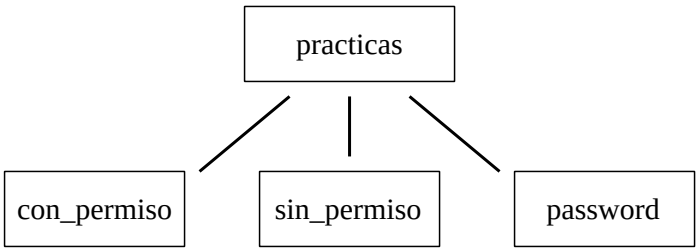

Figura 1.2: Estructura de directorios a crear.

3. Deduzca qué permisos (lectura, escritura, ejecución) es imprescindible que tenga como mínimo el propietario del directorio `sin_permiso` (es decir, usted) para poder realizar con éxito los siguientes comandos. Compruébelo modificando los permisos del directorio y ejecutando los comandos.

Comandos	permiso r	permiso w	permiso x
`cd sin_permiso`			
`cd sin_permiso; ls -l`			
`cp arbol.txt sin_permiso`			

Ejercicio 4

Comandos `tar` y `gzip`

1. Vamos a hacer una copia de seguridad del directorio practicas. Primero agrupe toda la estructura de directorio practicas en un único fichero llamado `seguridad.tar`.
2. Ahora comprima dicho fichero para que ocupe menos espacio. ¿Qué fichero se ha creado? ¿Cuánto se ha comprimido? Este fichero es una copia de seguridad del directorio practicas, que ocupa menos espacio. Ahora usted podría almacenar este fichero en un soporte magnético, o copiarlo en otro directorio, o llevarlo vía *ftp* a otra máquina.
3. Escriba la instrucción o instrucciones que usaría para recuperar toda la estructura del directorio original.

Ejercicio 5

Redireccionamiento del DISPLAY

1. Con la ayuda de los compañeros de la máquina de al lado, realice el siguiente ejercicio.
   Primero permita que cualquier máquina pueda abrir un *display* en su máquina local.
   Seguidamente mediante el comando `export`, indique como DISPLAY la máquina de sus compañeros.
   Ejecuten el comando `xeyes` para que se visualice en la máquina remota.