



UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

MANUAL DE PRACTICAS
TALLER DE SISTEMA OPERATIVO UNIX

ELABORADO POR

M.C FELICITAS PEREZ ORNELAS
M.I ALMA LETICIA PALACIOS GUERRERO

PRÁCTICA 1. INTRODUCCIÓN AL SISTEMA OPERATIVO UNIX

Sistema Operativo

Un sistema operativo es software que supervisa la forma en que se pueden usar los recursos de una computadora. En algunas computadoras el sistema operativo es un solo programa y en otras es un conjunto de programas que interactúan entre sí de diversas formas.

Funciones de un Sistema Operativo

- Provee la interfaz entre el usuario y la máquina.
- Llevar cuenta de fecha y hora. El hardware tiene un reloj integrado pero el sistema operativo lo lee y actualiza.
- Ejecución de la mayoría de las operaciones de entrada/salida y organización del disco.
- Provee acceso a los dispositivos de entrada/salida.
- Protección de archivos y datos.
- Permite a los usuarios compartir datos.
- Proporcionar herramientas.
- Proporciona mecanismos para la recuperación de errores
- Coordinar la secuencia de eventos.
- Asigna a los usuarios una parte justa de los recursos de la computadora entre los que se encuentran memoria, espacio de disco, tiempo de procesamiento, etc.

Para realizar sus funciones un sistema operativo está organizado en módulos. Estos son:

- Manejo de Memoria.
- Manejo de E/S.
- Manejo del Sistema de Archivos.
- Manejo de procesos.

Unix

Unix fue uno de los primeros sistemas operativos escritos en un lenguaje de programación de alto nivel, fue desarrollado en los laboratorios Bell, a finales de los 60's. Es un sistema multiusuario, multitarea y multiproceso. Fue diseñado para ser un sistema pequeño y flexible usado exclusivamente por programadores.

Historia de Unix

UNIX fue desarrollado originalmente por los laboratorios BELL de AT&T en 1969. Las regulaciones federales que existían en esa época le prohibieron entrar a la industria computacional y generar utilidades con las ventas de UNIX. Por esta razón AT&T distribuyó el sistema a un bajo costo entre colegas y universidades. Pronto se popularizó entre científicos y académicos. También se otorgaron licencias a otras compañías quienes desarrollaron sus propias versiones para utilización comercial. En 1980 AT&T tuvo libertad de comercializar Unix y a partir de entonces ha penetrado fuertemente en el mundo de los negocios.

Características de Unix

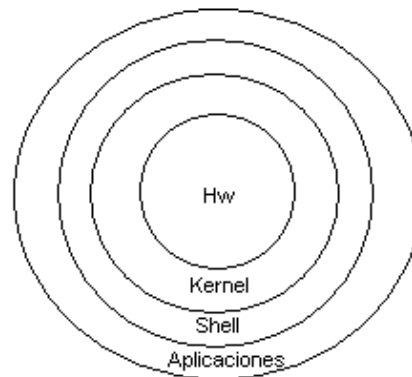
Multiusuario: Varias personas tienen acceso al sistema al mismo tiempo, compartiendo recursos, pero manteniendo algunos recursos como personales, por ejemplo archivos y directorios.

Multitarea: El procesador para ejecutar múltiples tareas al mismo tiempo. El procesador es un dispositivo mucho más rápido que muchos dispositivos conectados a la computadora, el sistema

operativo trata de mantener ocupado al procesador tanto como sea posible, haciendo un poco de trabajo para un usuario y luego para otro.

Multiproceso: Unix tiene la posibilidad de trabajar con dos o más procesadores conectados. Los sistemas con multiproceso pueden ejecutar instrucciones del mismo o de diferente programa al mismo tiempo.

Estructura de UNIX



Kernel: Es el núcleo del sistema operativo. Es el conjunto de software que proporciona las capacidades básicas del sistema operativo. Sus funciones son:

- Manejar la memoria de la computadora
- Controlar el acceso a la computadora
- Mantener el sistema de archivos
- Manejar interrupciones (señal para terminar ejecución)
- Manejar errores
- Realizar servicios de entrada y salida
- Asignar los recursos de la computadora

Shell: El shell es un programa que ejecuta otros programas. Se dice que “habla” con el usuario a nombre del sistema operativo. El shell lee la línea de comando que el usuario teclea, determina lo que significa e indica al kernel la ejecución de esos comandos.

- En algunos shells existen características que se pueden usar para reducir la escritura de nombres de archivos, comandos o rutas.
- Otros shells permiten asignar nombre cortos a los comandos.
- Los shells pueden llevar un registro de todos los comandos que se han usado recientemente, para que se les puede editar o reejecutar.
- Los shells permiten la ejecución de un conjunto de comandos contenidos en un archivo.

Algunos sistemas operativos solo reconocen un shell, pero Unix tiene la capacidad de usar un shell creado o adquirido en vez del estándar. Entre los shells más conocidos están:

- Korn Shell, interfase escrita por David Korn.
- Bourne Shell; viene incluido en UNIX que distribuye AT&T. La versión original de este shell fue desarrollada por Stephen Bourne en los Laboratorios Bell.
- C Shell, desarrollado en la Universidad de Berkeley por Bill Joy. Fue diseñado pensando en que los usuarios serían programadores de C.
- Bourne-Again Shell: bash

Al encender el servidor, el programa `init` se encarga de la inicialización de la máquina, creando la estructura que soporta los procesos multiusuario. Por cada puerto de terminal activo se inicia la ejecución de un programa `getty` que se encarga de establecer la velocidad de comunicación, tipos de terminal y modo. Luego, este mismo programa obtiene la cadena `login` que aparece en el terminal invitando al usuario a conectarse.

Una vez que el usuario introduce su nombre, `getty` llama al programa `login` enviándole el nombre de usuario como parámetro. El programa `login` se ocupa de comprobar si el nombre de usuario es válido y si el password es coincide. Si todo está correcto, llama al programa `sh` (shell) que se encarga a su vez de ejecutar los comandos que se encuentran en el archivo `.profile` en el directorio `HOME` de cada usuario. Finalmente aparece en pantalla el símbolo del shell (`$`). A partir de aquí, el shell se queda esperando a que se introduzcan comandos.

Cuando el usuario introduce un comando, el shell analiza la línea, verifica la sintaxis y lo ejecuta. El ciclo se repite hasta que el usuario se desconecta. Entonces, el programa `sh` termina su ejecución e `init` recobra el control iniciando una nueva ejecución de `getty` para la terminal.

Conceptos Básicos

Cuentas de usuario. Para ingresar al sistema, organizar y registrar las actividades de cada usuario, el sistema operativo proporciona y utiliza una cuenta por usuario. La cuenta de usuario contiene la siguiente información:

Login Name. Este es el nombre con el usuario será identificado en el sistema.

Password: Para mantener la seguridad del sistema, cada usuario debe tener una contraseña. Esta contraseña se introduce después del nombre de acceso, al intentar ingresar al sistema.

Group Identification. Cada usuario en el sistema es conocido individualmente y como miembro de un grupo. La pertenencia a un grupo es importante por razones de seguridad. Como miembro de un grupo, se permite el acceso a archivos y directorios a los que no se podría acceder en forma individual.

Home Directory. Este es el lugar en el sistema de archivos (Filesystem) donde se mantienen los archivos personales de cada cuenta de usuario. Al atarse al sistema, cada usuario es direccionado a su directorio de casa.

Super Usuario. Además de tener cuentas de usuario individuales, cada sistema UNIX tiene una cuenta de "superusuario", conocido también como "root". Para la realización de tareas de administración del sistema, el administrador del sistema debe acceder al mismo como superusuario. El superusuario puede leer y editar cualquier archivo en el sistema, así como ejecutar cualquier programa.

Actividades:

1. Abrir una sesión de trabajo en el servidor Sun205. La dirección es 148.231.130.230
2. Introduzca su login. (al seguido de los últimos 6 dígitos de su matrícula)
3. Introducir password. Por ser la primera vez se pedirá que escriba el password dos veces. El password debe apegarse a las siguientes reglas.
 - Longitud de al menos seis caracteres
 - Al menos un caracter debe ser en mayúscula o no alfabético
 - El password nuevo debe ser diferente al password
 - No podrá ser igual al nombre del usuario
4. Terminar sesión.

PRÁCTICA 2 SISTEMA DE ARCHIVOS

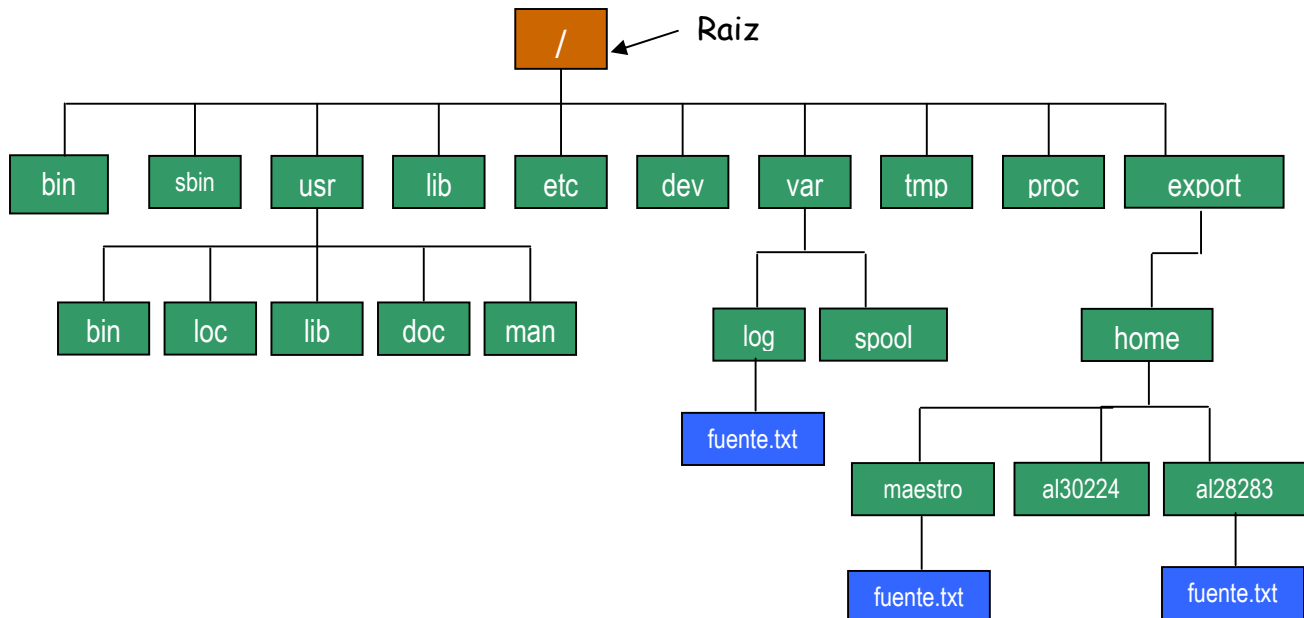
En UNIX todos los programas, datos, directorios y dispositivos son archivos. Un archivo es una sucesión de bytes. El sistema de archivos de UNIX está organizado en una jerarquía de directorios que tiene una forma arborescente.

- Jerárquico: El sistema de archivos es un árbol de directorios.
- Virtual: Los archivos representan objetos como unidades de disco, impresoras, etc u objetos lógicos como procesos, enlaces con otras partes del sistema de archivos.
- Cada archivo en el directorio se representa internamente por un i-nodo. Un i-nodo es una estructura de datos que contiene la siguiente información: fecha de modificación, dueño, tamaño y permisos.

El sistema de archivos tiene una serie de directorios estándar:

/	Directorio raíz
/bin	Comandos básicos de usuario
/sbin	Comandos básicos de superusuario
/usr	Aplicaciones e información sobre el sistema
/usr/bin	Comandos de usuario
/usr/local	Información y comandos instalados específicamente en la máquina
/usr/lib	Librerías de aplicaciones
/usr/doc	Documentación sobre aplicaciones
/usr/man	Páginas de manual
/lib	Librerías del sistema
/etc	Archivos de configuración del sistema
/dev	Archivos de dispositivos del sistema
/var	Archivos de trabajo del sistema
/var/log	Información sobre operaciones del sistema: accesos, mensajes, avisos de seguridad, etc,
/var/spool	Archivos de colas de impresora, correo, etc.
/tmp	Archivos temporales del sistema o de usuarios
/proc	Representación de procesos
/home	Directorios de usuarios

El árbol de directorios de Unix puede representarse como un árbol invertido:



En Unix, a la parte superior del árbol de directorios se le conoce como raíz.

Directorio padre. Cuando un directorio contiene a otros directorios o archivos se dice que es su directorio padre. Por ejemplo en la figura observamos que el directorio export es padre del directorio home y este a su vez, es padre de al30224 y al28283.

Directorio actual. El directorio actual es el punto del árbol de directorios se este trabajando. Al entrar al sistema el usuario siempre se encontrará en su **home directory**.

- . Un punto representa al directorio actual.
- .. Dos puntos representan al directorio padre del directorio actual.
- / La diagonal representa a la raíz.

Rutas Absoluta y Relativa

Una ruta es el camino a seguir en el árbol de directorios para localizar un archivo o un directorio. Las rutas pueden ser absolutas o relativas.

Ruta Absoluta. Cuando la secuencia de directorios se escribe empezando con / (root), entonces la búsqueda del archivo o directorio será desde la raíz del árbol de directorios. Las rutas absolutas son un mapa de localización de archivos y son únicas en el sistema. La siguiente línea es un ejemplo de ruta absoluta.

`/export/home/maestro/fuente.txt`

Ruta Relativa. Se le llama ruta relativa porque depende del directorio de trabajo actual. La secuencia toma como punto de partida el directorio actual.

```
./fuente.txt
../home/maestro/fuente.txt
./log/fuente.txt
```

Comandos básicos para el manejo de directorios en UNIX

Comando en UNIX	Comando en DOS	Función
ls	dir	Muestra el contenido del directorio
mkdir	md	Crear un directorio nuevo
rmdir	rd	Borrar un directorio existente
pwd	No existe	Directorio actual de trabajo
mv	ren	Renombra un directorio
cd	cd	Cambio de directorio
uname		Muestra información sobre el sistema UNIX instalado en el servidor.

Formas de uso de los comandos

pwd

pwd ↵ Muestra el directorio de trabajo actual. Al ejecutar la línea: \$pwd↵
La salida que se observa es: \export\home\maestro

Se recomienda que consulte el directorio de trabajo antes de realizar cualquier operación.

cd cambia al directorio que se indique (si es que existe)

```
cd ...↵ Regresa al directorio padre.
cd <directorio>↵ Cambia el directorio de trabajo al directorio especificado.
cd ↵ Cambia al home directory del usuario.
```

ejemplo:

```
$cd al30224↵
```

ls Muestra el contenido del directorio actual.

ls -F ↵ Muestra el contenido del directorio actual anteponiendo un símbolo al nombre cada archivo, para indicar el tipo de archivo es. Una / indica directorio y un * archivo ejecutable.

ls -R ↵ Lista el directorio de trabajo así como también todos los subdirectorios.

ls -l ↵ Listado con detalles en orden alfabético.

ls -a ↵ Muestra los archivos ocultos.

ls -r ↵ Muestra el contenido de un directorio en sentido inverso.

\$ls -la ↵

Las opciones del comando se pueden combinar, el ejemplo siguiente muestra todos los detalles de los archivos incluyendo los archivos ocultos.

\$ls -la ↵

mkdir Crear directorios

mkdir <nuevo directorio> ↵ Crea un nuevo directorio en el directorio actual.

\$mkdir tareas ↵

mv El comando mv tiene dos aplicaciones renombrar un directorio o moverlo hacia otra parte.

mv <nombre anterior> <nombre nuevo> ↵ Renombra el directorio con el nombre nuevo.
ejemplo:

\$mv agenda2002 agenda2003 ↵

mv <fuente> <destino> ↵ Mueve un directorio a otra parte del árbol de directorios. Por ejemplo:

\$mv tareas ./taller/

En la línea anterior se usa una **ruta relativa**, el comando mueve el archivo tarea.doc del directorio actual al directorio taller que se supone está dentro del directorio actual.

\$mv tareas /bin/

con una **ruta absoluta**, mueve el directorio tarea del directorio actual al directorio bin que está en la raíz

rmdir Borra un directorio

rmdir <directorio> ↵ Borra el directorio especificado, siempre y cuando esté vacío.

\$rmdir tareas

Actividades:

1. Despliegue el nombre del directorio de trabajo actual.
2. Lista en forma de columnas (sin detalles) el contenido del directorio padre de su **home directory**.
3. Lista en orden alfabético inverso todos los archivos (incluyendo los ocultos) de su **home directory**.
4. Lista en orden alfabético el contenido de su **home directory** mostrando información detallada. ¿En qué consiste esa información? ¿Qué significa el primer caracter que se muestra en la lista?
5. Desarrolle la estructura de directorios que se indique en el pizarrón.
6. Verifique que la estructura haya sido creada correctamente.
7. Borre el último nivel del árbol de directorios.
8. Lista el contenido de su directorio, mostrando de forma simbólica el tipo de archivos que contiene.
9. ¿Para qué sirve el comando **whoami**?
10. ¿Qué información nos proporciona **uname**?
11. Dentro de un directorio llamado **alumnos**, cree un directorio para cada alumno del salón, asignándole como nombre el user name de cada persona (verifique la lista de usuarios mediante el comando **who**).
12. Renombre todos los directorios del directorio **alumnos** con los nombre reales de sus compañeros.
13. Liste los directorios en forma alfabética. Quién es el dueño de los directorios creados?, ¿Cuál es la fecha de creación?
14. Borre en un solo paso la estructura anterior. Auxiliense del manual de ayuda.

PRÁCTICA 3. ADMINISTRACIÓN DE ARCHIVOS

Un **archivo**: Es una colección de bytes. Constituye la unidad fundamental de un sistema de archivos en Unix. Cada archivo tiene los siguientes atributos:

- Un nombre de archivo. No necesariamente único en el sistema, pero si en el directorio.
- Un número de filesystem único, conocido como i-node.
- Un tamaño en bytes.
- La hora de última modificación.
- Un juego de permisos de acceso.
- Un dueño.
- Un grupo.

Tipos de Archivo en Unix

Existen tres diferentes tipos de archivos en Unix: Archivos ordinarios, archivos de dispositivo y archivos de directorio.

Archivos ordinarios. Generalmente son documentos, códigos fuente de programas, o datos de programas. Los archivos binarios ejecutables (programas) se consideran también archivos ordinarios. Los bytes de un archivo ordinario se interpretan como caracteres texto, instrucciones binarias, o cláusulas de programas, por los programas que los examinan.

Archivos de Dispositivos. Cada dispositivo físico en el sistema, tales como un disco duro, disco flexible, impresores, terminales y el sistema de memoria tienen asignados un archivo especial. Estos archivos son llamados archivos de dispositivos.

Archivos de Directorios. Los archivos de directorios son los lugares donde los archivos son almacenados (conceptualmente, no físicamente). Un archivo de directorio es referido como un "directorio" y contiene los nombres y la localización de los archivos "que están en él".

Nombres de archivos

Un nombre de archivo es una secuencia de caracteres consistente de letras, dígitos y caracteres especiales. Los nombres de archivo deben indicar el contenido de los mismos. Estos nombres deben ser únicos en el directorio y pueden repetirse en todo el sistema. Directorios diferentes pueden contener diferentes archivos con el mismo nombre. Cuando un nombre de archivo contiene un punto al inicio (.), es un archivo "oculto." Los archivos de configuración del sistema por lo general son archivos ocultos. Los caracteres como ?, *, [,], y guión nunca deben usarse para nombrar archivos porque tienen un significado especial para el shell.

Comodines

Un comodin es un caracter que el shell usa para representar uno o mas caracteres del nombre de uno o mas archivos. Unix emplea los siguientes comodines: *, ?, []

- * Equivale a cualquier conjunto de caracteres de nombre de archivo.
- ? Coincide con un solo carácter cualquiera de nombre de archivo
- [] Coincide con una clase de posibles caracteres de nombre de archivo.

Comandos para el manejo y administración de archivos

Comando	Función
cat	Despliega el contenido de un archivo, Crea un archivo.
more	Muestra el contenido de un archivo haciendo pausas.
touch	Cambia la fecha y hora de creación, modificación o última lectura de un archivo.
mv	Renombra un directorio.
cp	Copiar archivos.
wc	Muestra cuántas palabras, caracteres y líneas que tiene un archivo.
tail	Muestra las últimas n líneas de un archivo.
head	Muestra las primeras n líneas de un archivo.
chmod	Permite cambiar los permisos de un archivo.

Formas de uso de los comandos

cat

cat <archivo> ↵ Lista el contenido de un archivo, no hace pausas.
 cat -n <archivo> ↵ Muestra el contenido del archivo, numerando cada línea.

```
$cat lista.txt↵
$cat -n lista.txt
```

more

more <archivo1, archivo2,...,archivon> ↵ Muestra el contenido de *n* archivos texto.

more [+líneas] [-inicio] <archivo> ↵ Muestra n líneas de un archivo a partir de la línea de inicio.

Teclas de control dentro de more

Barra espaciadora	avanzar una página.
Enter	avanzar una línea.
n	avanzar al siguiente archivo
q	salir de more

touch

touch [-t MMDDhhmm] <archivo> ↵ Cambia la fecha y hora del archivo. Donde
 MM=días DD=días hh=horas mm=minutos.

```
$touch -t 01011212 carta.txt↵
```

mv Renombre o cambia de directorio un archivo

mv <nombre viejo> <nombre nuevo> ↵ Renombrar archivo.
 mv <destino1 destino2 ... destinon> <destino> ↵ Mueve los archivos al directorio destino.

Si agrega la opción -i pregunta antes de sobrescribir el archivo.

```
$mv -i tarea nuevo↵
```

cp

`cp <fuente> <destino>` Copia el archivo fuente al archivo destino
`cp <fuente1 fuente2 fuente3 ...fuenteN> <directorio destino>` Copia los archivos al directorio destino.

Si agrega la opción `-i` pregunta antes de sobrescribir el archivo. Ejemplo:

`$cp -i pract1 ./areas/unix/`

tail Muestra las últimas líneas de un archivo, por omisión se muestran las últimas 10.

`tail <archivo>` Muestra las últimas 10 líneas del archivo.
`tail -n <archivo>` Muestra las n últimas líneas del archivo.

`$tail -7 pract1`

head Muestra las primeras líneas de un archivo, por omisión se muestran las primeras 10.

`head <archivo>` Muestra las primeras 10 líneas del archivo.
`head -n <archivo>` Muestra las n primeras líneas del archivo.

`$head -15 pract1`

wc Cuenta las palabras, líneas y caracteres que tiene el archivo.

`wc <archivo>` Muestra cuantos caracteres, líneas y palabras tiene el archivo.
`wc -c <archivo>` Muestra el total de caracteres que tiene el archivo.
`wc -w <archivo>` Muestra el total de palabras que tiene el archivo.
`wc -l <archivo>` Muestra el total de líneas que tiene el archivo.

`$wc -wl pract1`

Permisos de archivos y directorios.

UNIX permite al dueño de un archivo o directorio restringir el acceso a ellos. Los permisos en un archivo limitan la lectura, escritura y/o ejecución, mientras que para un directorio limitan a quien pudiera cambiarse a ese directorio, listar su contenido así como crear y borrar archivos dentro del mismo.

drwx r-x r-x 2 maestro staff 512 Mar 3 15:31 ejemplos

Permisos para dueño
 Permisos para grupo
 Permisos para otros usuarios.

De izquierda a derecha, los caracteres se interpretan como 3 juegos de permisos. Cada uno establece los siguientes permisos:

Para los archivos ordinarios, los permisos tienen el siguiente significado:

r El archivo puede leerse.
w El archivo puede editarse
x El archivo puede ejecutarse.
- El permiso no está otorgado o nulo.

Para los directorios, los permisos tienen el siguiente significado:

- r** Los archivos pueden listarse, el directorio además requiere el permiso de ejecución "x".
- w** Pueden crearse o borrarse archivos en el directorio.
- x** Puede buscarse en el directorio.
- El permiso no está otorgado o nulo.

Cambio de los permisos de archivos.

El comando **chmod** cambia los permisos de lectura, escritura y ejecución y busca permisos en un archivo o directorio. La sintaxis es la siguiente:

chmod <modo><archivo>

Hay dos métodos para usar el comando chmod, uno de ellos se vale de números y se llama método absoluto; el otro utiliza símbolos y recibe el nombre de método simbólico.

Método Absoluto. Este método, también conocido como numérico, usa un número octal de tres dígitos para almacenar los permisos.

Permiso	Símbolo	Valor octal
Lectura	r	4
Escritura	w	2
Ejecución	x	1

Para encontrar los dígitos octales que necesita para especificar los permisos para una cierta categoría (usuario, grupo u otros), basta con sumar los números que estén asociados con los permisos que desee activar. Por ejemplo: Suponga que se desea modificar los permisos del archivo pract1 de la siguiente forma: activar todos los permisos para el usuario, solo lectura y ejecución para el grupo y ninguno para otros usuarios. La asignación de permisos sería:

Dueño			Grupo			Otros		
r	w	x	r	w	x	r	w	x
✓	✓	✓	✓	x	✓	x	x	x
4	2	1	4	0	1	0	0	0
7			5			0		

El comando chmod para otorgar estos permisos se escribiría: **chmod 750 pract1**

Método simbólico. Utiliza símbolos para establecer categorías y permisos

Los usuarios se especifican como sigue:

- u** Usuario, el dueño de un archivo o directorio.
- g** Grupo, el grupo de usuarios al cual el dueño del archivo pertenece.
- o** Otros, todos los usuarios del sistema que no están en u o g.
- a** Todos los usuarios del sistema.

Ejemplos de cambio de permisos:

- chmod u+x historia** Otorga el permiso de ejecución al dueño del archivo.
- chmod go+x tareas** Otorga al grupo y a otros usuarios el permiso de ejecución sobre el archivo.
- chmod o-w alumnos** Prohíbe a otros usuarios la escritura en el archivo.
- chmod o+r-wx uno** Otorga permiso de lectura para otros usuarios. Suprimen los permisos de escritura y de ejecución.

Actividades:

1. Copie el archivo primero que está dentro de maestro/poesia a su home directory.
2. Copie el archivo segundo (está en el mismo directorio) home directory.
3. Copie el archivo intermedio de maestro/poesia home directory.
4. Verifique el contenido de los tres archivos.
5. Borre los tres archivos copiados en los pasos 2, 3 y 4.
6. Copie los tres archivos (primero,segundo e intermedio) utilizando una sola instrucción.
7. Muestre en pantalla el contenido de segundo e intermedio usando una sola línea.
8. Muestre en pantalla el contenido de Enpaz.txt, numerando cada línea.
9. Copie el archivo SuavePatria.txt de maestro/poesia a su directorio.
10. Muestre en pantalla el contenido del archivo amorosos.txt en el monitor.
11. Vuelva a mostrar el archivo amorosos.txt, pero por páginas.
12. Muestre las últimas diez líneas de este archivo
13. El archivo SuavePatria.txt ahora se llamara patria.
14. Muestre las últimas ocho líneas del archivo amorosos.txt.
15. ¿Cuántas palabras en total contiene el archivo amorosos.txt ?
16. ¿Cuántos caracteres en total contiene el archivo Enpaz.txt ?
17. Mostrar los permisos de todos los directorios que están en el directorio home. Observe cómo están los permisos para grupo y otros usuarios.
18. Otorgue permiso a su grupo para leer y escribir en su directorio.
19. Seleccione a uno sus compañeros, escriba un archivo en su directorio, llamado películas (use el comando cat). Escriba un párrafo sobre la última película que haya visto en el cine, puede ser una sinopsis o su opinión personal.
20. Otorgar permiso de lectura al grupo para este archivo.
21. Copia la historia de tres de tus compañeros a un directorio llamado **sinopsis**.
22. Restringir los permisos de lectura y escritura de tu directorio al grupo.

PRÁCTICA 4 FILTROS Y EXPRESIONES REGULARES

Filtro: Es cualquier programa que toma sus datos de la entrada estándar (stdin) y muestra sus resultados en la salida (stdout). Unix tiene varios filtros que permiten seleccionar la información contenida en un archivo de acuerdo a criterios establecidos con expresiones regulares.

Expresión Regular: Una expresión regular en Unix se compone de forma similar a una expresión aritmética. La unidad mínima para construir una expresión regular es un carácter. Los números y las letras se representan a sí mismos; existen algunos caracteres que se combinan para formar patrones. A continuación se listan algunos ejemplos:

Patrón	Significado
casa	Cadena casa
^Luna	La cadena se buscara al inicio de la línea
Gato\$	La cadena se buscará al final de la línea.
^Sol\$	La cadena se buscará como línea única
.(punto)	Cualquier caracter
[Gg]ato	Busca las cadenas gato o Gato
[Cc][Aa]sa	Busca las cadenas Casa,CASA,casa cAsa
.erro	Busca cualquier cadena que termine con erro. (berro, perro)
^[^Gg]	Busca las líneas que no empiecen con G ni con g
[a-d]ato	Busca aato,bato,cato,dato
[a-zA-D]ato	Busca aato, Aato,bato,Bato, cato,Cato,Dato ,dato

sort

sort -f <archivo> ↵	Ordena considerando de igual valor mayúsculas y minúsculas.
sort -M <archivo> ↵	Compara considerando los tres primeros caracteres de la línea como el nombre de un mes en inglés.
sort -n <archivo> ↵	Ordena en forma numérica ascendente.
sort +1 <archivo> ↵	Ordena por la segunda columna. (+2 por la tercera, etc), considera como delimitador el espacio y el tabulador.
sort -r <archivo> ↵	Invierte el orden.

grep Global Regular Expression and Print

grep <cadena> <archivo> ↵	Muestra la línea(s) donde encuentra la cadena.
grep -n <cadena> <archivo> ↵	Muestra la línea y el número de línea en donde encuentra la cadena.
grep -c <cadena> <archivo> ↵	Muestra cuántas líneas contienen el patrón especificado.
grep -v <cadena> <archivo> ↵	Muestra las líneas que no cumplen con el patrón de búsqueda.
grep -w <cadena> <archivo> ↵	Muestra las líneas que contienen la cadena como palabra completa.
grep -w <'frase'> <archivo> ↵	Muestra la línea donde se encuentra la frase completa.
grep -i <cadena><archivo> ↵	Evita la distinción entre mayúsculas y minúsculas.

cut

<code>cut -f<numero> <archivo>.</code>	Selecciona sólo el campo n. Por default, el delimitador entre columnas es el tabulador.
<code>cut -f<inicio-fin> <archivo>.</code>	Selecciona el rango de columnas desde inicio a fin.
<code>cut -f<col1,col2,coln> <archivo>.</code>	Selecciona sólo las columnas especificadas en la lista.
<code>cut -d'caracter' -f<col> <archivo>.</code>	Muestra la columna indicada, considerando como delimitador entre columnas el caracter especificado en caracter.
<code>cut -c<columna> <archivo>.</code>	Muestra de cada renglón el carácter indicado.
<code>cut -c<inicio-fin> <archivo>.</code>	Muestra de cada renglón los caracteres inicio a fin. Por ejemplo:1 al 10.
<code>cut -c<col1,col2,coln,> <archivo>.</code>	Muestra de cada renglón las columnas seleccionadas.

Actividades:

1. Cree un directorio llamado filtros en su home directory
2. Elimine todos los permisos de este directorio, solamente usted tendrá todos los permisos.
3. Del directorio maestro/filtros copie el archivo nombres a su directorio filtros
4. Muestre el archivo nombres ordenado en forma alfabética ascendente por apellido materno.
5. Copie el archivo matriculas de maestro a su directorio filtros
6. Ordene el archivo matriculas por el apellido materno.
7. Ordene el archivo nombres por matricula
8. ¿Qué hace el comando `sort -f <archivo>?` (Lea el manual en línea)
9. Ordene el archivo matriculas en forma descendente por apellido paterno.
10. Copie a filtros el archivo delim.
11. Copie a filtro el archivo maestro/filtros/lista
12. Ordene el archivo matriculas en forma numérica descendente.
13. Ordene el archivo nombres por el nombre en forma descendente.
14. Mostrar la primera y ultima columna del archivo asteriscos.
15. ¿Qué grupo tiene mas alumnos que se llaman Jose lista o nombres?
16. Muestre a todos los alumnos cuya matricula empiece con 55 en el archivo nombres (son 8)
17. Cuantas líneas tiene el archivo hormigas? (está en maestro/)
18. Muestre en qué líneas está la frase "las hormigas" en el archivo hormigas
19. Muestre solo las líneas que empiezan con "La" en hormigas
20. Mostrar a los alumnos que tiene un nombre o apellido que tiene cuatro letras en total.
21. Mostrar todos los alumnos que no se apellidan Hernandez en lista.
22. Mostrar solamente el nombre y apellido del archivo delim.
23. Mostrar solamente las matriculas del archivo delim.
24. Muestre en pantalla sólo la primer columna de caracteres del archivo lista
25. Muestre ahora los caracteres 18 al 25 de hormigas
26. Muestre el campo número 3 del archivo algoritmos2.
27. Muestre los renglones del archivo lista donde el apellido paterno sea Hernandez.
28. Muestre los renglones del archivo delim cuyo nombre sea luis.
29. Muestre las líneas del archivo hormigas que contienen la palabra Utopia. ¿Qué número son?
30. Muestre la línea de lista que contiene el nombre Noemi al final.
31. Muestre las líneas del archivo hormigas que contiene las palabra hormigos u hormigas.
32. Mostrar las líneas que no empiezan con F-G-H en asteriscos.

PRÁCTICA 5 REDIRECCIONAMIENTO DE ENTRADA - SALIDA

En el sistema Operativo Unix cada comando o programa en ejecución está relacionado con tres flujos: flujo de entrada, de salida y flujo de error.

Standard input (stdin): Es el flujo desde el cual la mayoría de los programas de UNIX toman sus datos de entrada, por lo general la línea de comando.

Standard output (stdout): Es el flujo hacia el cual la mayoría de los programas envían sus resultados, normalmente es la pantalla.

Standard error (stderr): Este flujo se utiliza para enviar información de depuración y errores, generalmente va hacia la pantalla.

En Unix, la implementación física y la organización lógica de un archivo son independientes; físicamente se accede a los archivos como bloques que están dispuestos en forma aleatoria. Lógicamente, todos los archivos están organizados como un flujo continuo de bytes. Esta organización se extiende a las operaciones de entrada y salida y permite que puedan ser redireccionados; es decir, cambiar el dispositivo al que normalmente están ligados.

Redireccionamiento

El redireccionamiento consiste en una capacidad que permite mover datos fácilmente hacia dentro/fuera de los archivos. Es posible redireccionar la salida estándar para que en lugar de verla por pantalla, quede guardada en un archivo; también se puede redireccionar la entrada estándar, desligándola del teclado para que la lectura de datos se efectúe desde un archivo en vez del teclado.

Redirección de la salida estándar

Si se deseara desviar la salida estándar a un archivo en vez de la pantalla, se escribe el operador de redirección de salida (>) y el nombre de un archivo en la línea de comando. La operación de redirección crea el nuevo archivo de destino. Si el archivo ya existe, su contenido será reemplazado por los datos de la salida estándar. Aunque el operador de redirección y el nombre de archivo se colocan detrás de la orden, la operación de redirección no se ejecuta después de la orden, sino antes; es decir, la orden que genera la salida es ejecutada sólo después de que el archivo de redirección haya sido creado.

Adición de la salida estándar. También puede agregar la salida estándar a un archivo existente usando el operador de redirección >>. En lugar de sobrescribir el archivo, los datos de la salida estándar se añaden al final del mismo.

Redirección	Ejecución
Comando > nombre archivo	Redirecciona la salida a un archivo o dispositivo, creando el archivo si no existe y sobrescribiéndolo si ya existe
Comando >> nombre archivo	Redirecciona la salida estándar a un archivo o dispositivo, añadiendo la salida al final del archivo.

Ejemplos:

\$ls -l > lista	Ejecuta el comando ls y envía el resultado al archivo lista.
\$finger >> usuarios	Ejecuta el comando finger y agrega el resultado al final del archivo usuarios.
\$cat arch1 arch2 arch3 > tmp	Copia el contenido de arch1, arch2 y arch3 en un solo archivo. La salida no se ve en pantalla.

Redirección de la entrada estándar

Muchos comandos necesitan recibir datos de la entrada de datos estándar. La entrada estándar recibe los datos de un dispositivo o de un archivo, por default la entrada de datos estándar es el teclado, los caracteres que son introducidos desde el teclado son colocados en la entrada estándar, que es entonces dirigida al comando de Linux. De manera similar a lo que sucede con la salida estándar, también se puede redirigir la entrada estándar, para recibirse desde un archivo en lugar desde el teclado. El operador para redirigir la entrada estándar es el **<**.

Redirección	Ejecución
Comando < archivo	Redirecciona la entrada desde un archivo o dispositivo hacia un comando.

Ejemplos:

\$ cat < lista

\$ cut -f1 <lista

Nota: \$ cat <<lista No tiene sentido

Puede combinar las operaciones de redirección tanto para la entrada como para la salida estándar. Observe el siguiente ejemplo, normalmente **cut** recibe entrada desde la entrada estándar y envía la salida a la salida estándar; pero en el ejemplo recibe la entrada desde un archivo y dirige la salida hacia otro archivo.

\$ cut -d':' -f2 <delim >salida

Comando tr

Uno de los comandos que requiere de redireccionamiento de entrada es **tr**, este es un comando que da como salida una "traducción" de la entrada, en la cual los caracteres pueden ser eliminados o sustituidos por otros.

Opciones de tr

tr "a" "A" < archivo

Sustituye el caracter a minúscula por A mayúscula.

tr "[a-z]" "[A-Z]" < archivo

Especifica un rango de caracteres (a-z) que son sustituidos por otro rango de caracteres (A-Z), en este caso sustituye minúsculas por mayúsculas.

tr -d "abc" <archivo

Suprime cualquier caracter dentro de la lista.

tr -s "." "\013" <archivo

Elimina caracteres repetidos. En este caso uno o mas puntos seran sustituidos por un enter.

tr -c

Complementa el conjunto que se va a traducir; es decir, invierte el sentido de la conversión. Se usa combinado con -s.

El comando **tr** reconoce expresiones de la forma **[:name:]** donde name tiene un valor dado en la categoría **LC_CTYPE**.

Expresión	Significado
lower	minúsculas
upper	mayúsculas
digit	dígitos (0-9)
space	espacio
punct	signos de puntuación (.,,:)
alnum	Letras y Numeros
alpha	Letras

Ejemplo: **tr -cs "[:digit:]" "[x*]" <archivo**

Sustituye todo lo que NO sea dígito por una x.

Entubamientos o canalizaciones

A menudo se encontrará con situaciones en las que tendrá la necesidad de enviar datos de un comando a otro y no a un archivo destino; es decir, necesitará **enviar la salida estándar de un comando hacia la entrada estándar de otro**. Para formar una conexión como ésta en Unix lo que se conoce como canalización o entubamiento. El operador de entubamiento es el pipe (|). Este se escribe entre dos comandos para establecer la conexión entre ellos, la salida del primero se convierte en la entrada para el siguiente.

Ejemplos:

\$cat -n carta more	El comando cat -n numera las líneas del archivo carta, la salida generada se muestra por pantallas con el comando more.
\$ls -l more	ls -l lista el contenido del directorio actual, por pantallas.

Comando tee

Copia la entrada estándar en un archivo y la envía a la salida estándar. Generalmente se usa como un filtro y permite guardar la salida en un archivo al mismo tiempo que se envía a otro comando. Por ejemplo:

sort original | tee salida

El resultado de sort se observa en la pantalla al mismo tiempo que se envía al comando tee para que genere un archivo cuyo contenido es el resultado observado en la pantalla.

\$sort archivo | cat -n | tee archivo_nuevo | lpr Guarda la salida de un archivo y lo imprima

Actividades:

1. Muestre el archivo lista con las líneas numeradas.
2. Muestre solamente a los alumnos que están reprobados (del mismo archivo).
3. Repita el paso anterior pero ahora redireccione la salida para generar un archivo llamado reprobados.
4. Muestre y genere un archivo con los artistas que graban con Sony (del archivo billboard).
5. Repita el paso anterior pero ahora debe verse en pantalla el resultado al mismo tiempo que se genera el archivo.
6. Genere un archivo con las últimas tres líneas del archivo diario.
7. Genere una lista ordenada y numerada de las personas aprobadas en los archivos lista y prografinal. Grabelo en el archivo reporte
8. El archivo reporte debe estar escrito todo en mayúsculas.
9. Agregue una fecha importante al archivo fechas.
10. Ordene el archivo fechas por mes, mostrando el contenido al mismo tiempo que genera el archivo calendario.
11. Cree el archivo nombres con solo el primer nombre de las personas aprobadas.
12. En el archivo de reprobados sustituya todos los ":" por espacios.
13. Genere un archivo que contenga solamente las matriculas de los alumnos aprobados en el archivo lista.
14. En el archivo lista sustituya todas las matriculas por algun otro carácter.
15. Genere un archivo con las primeras 20 lineas del archivo diario.
16. Sustituya todos los simbolos de puntuacion en diario por otro carácter.
17. Genere un archivo con todas las personas que están trabajando en el sistema actualmente, pero la salida debe de verse en pantalla al mismo tiempo que se genera el archivo.
18. Genere un archivo de su directorio con detalles, ordenado por numero de i-nodo. El resultado además debe verse en pantalla.
19. Genere un archivo que contenga las lista de usuarios del sistema que pertenecen al grupo unix632, esten o no actualmente en el sistema.
20. Genere un archivo que contenga el user name y grupo de los usuarios cuyo login name empieza con al.

Preguntas:

Investigar y explicar el uso del comando diff

PRÁCTICA 6. COMUNICACION ENTRE USUARIOS

Introducción

El sistema operativo UNIX provee varias maneras para que un usuario se comunique con otros usuarios. Podemos encontrar dos tipos de comunicación:

- 1.comunicación interactiva
- 2.comunicación no interactiva

La comunicación no interactiva es aquella en la que se emplean archivos para guardar los mensajes que se desean enviar a un o más usuarios. El correo electrónico es la aplicación que nos permite realizar esta tarea.

La comunicación interactiva es aquella en la que la comunicación solo puede establecerse si el (los) otro(s) usuario(s) tiene(n) una sesión activa en el sistema en el momento en el que se desea comunicar.

Comandos para Comunicación No Interactiva

mailx

Dentro de casi cualquier sistema Unix disponemos de un programa básico de email en consola (modo texto) llamado Mailx que nos permitirá el envío, lectura y contestación de mensajes de correo electrónico.

Envío de un correo

La sintaxis de mailx es: **mail [usuario(s)]**

\$ mailx maestro ↵

Al introducir la línea anterior se pide que escriba el subject: del correo. Si no desea escribirlo presione la tecla <enter>

Subject: prueba ↵

A continuación viene la sección donde se escribe el mensaje. Para finalizar el mensaje escriba una línea que contenga solamente un punto.

**esto es un mensaje de prueba
.** ↵

Una finalizado el texto del mensaje aparecerá el reconocimiento del final de mensaje EOT (End Of Text). y el mensaje será enviado.

EOT

Si desea cancelar el mensaje, presione <control><c>. El texto se guardará en el archivo dead_letter.

El mensaje también puede estar escrito en un archivo de texto. En este caso el procedimiento sería

\$mail destinatario < archivo

Ejemplo: `$ mailx maestro <mensaje`

Revisar y Contestar Correo

Si existe correo pendiente en el buzón, al iniciar la sesión aparecerá el mensaje “You have new mail” .

Para revisar el correo se escribe el comando mailx. Este nos mostrará una lista de todos los mensajes en el buzón, indicando el estado del mensaje (N=New, U=Unread). Aparece también un número que indica el orden de llegada, el remitente, la fecha de envío y el asunto.

```
$ mailx
mailx version 5.0 Sat Apr 6 14:57:29 PST 2002 Type ? for help.
"/var/mail/maestro": 2 messages 2 new
>N 1 lety@Sun205    Fri Apr 11 20:31 17/556 prueba
  N 2 lety@Sun205    Fri Apr 11 20:31 17/568 prueba2
?
```

mailx tiene sus propios comandos y su propio símbolo de línea de comando, el signo de interrogación. Entre los comandos básicos de mailx se encuentran:

Comandos para leer correos recibidos

? <enter>	Muestra el primer mensaje recibido
?<Número>	Muestra el mensaje que tenga el número especificado

Comandos para borrar correos recibidos

d <enter>	Borra el correo que se este mostrando actualmente
d<Número>	Borra el correo especificado

Comandos para responder mensajes

r	Responde el correo que se este mostrando actualmente
r<Número>	Responde el correo especificado

Comandos para salir

x	sale dex mail sin eliminar los mensajes borrados con el comando d
q	sale del mail eliminando todos los mensajes marcados.

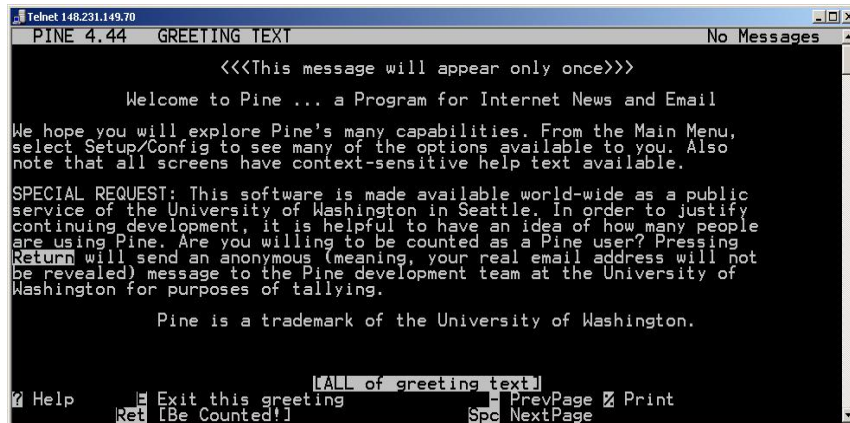
PINE (Program for Internet News & Email)

Pine es una herramienta para lectura, envío y administración de mensajes electrónicos, fue desarrollado por el grupo Computing & Communications en la Universidad de Washington. Aunque originalmente fue diseñado para usuarios sin experiencia en correo electrónico, pine ha evolucionado y ahora tiene muchas características avanzadas. Su distribución es gratuita.

Acceso a Pine

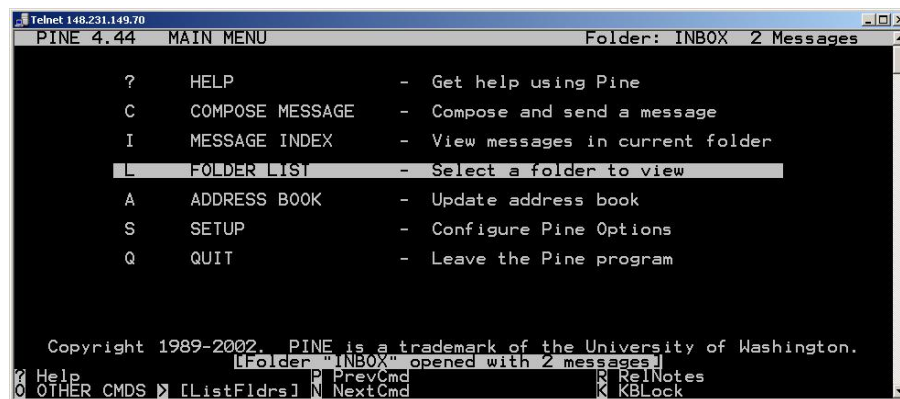
`$pine`

La primera vez que haga uso de pine observará una pantalla como la siguiente:



Para avanzar hacia el menú principal presione la tecla <E>

Menú Principal



Desde el Menú Principal, se puede consultar la ayuda, escribir y enviar un mensaje, ver la lista de correos, abrir directorios, actualizar la agenda, configurar las opciones de Pine y salir del Pine.

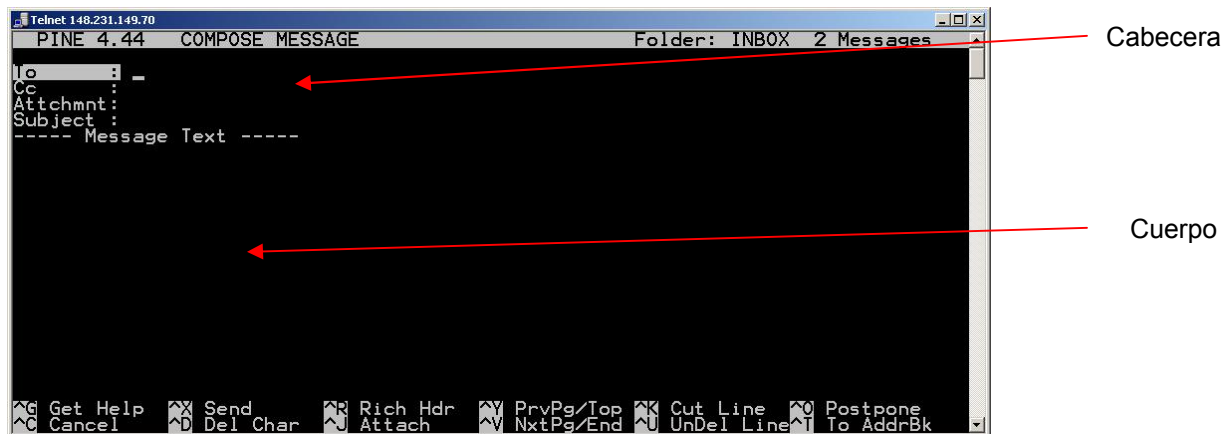
En la parte inferior de la pantalla se muestra un menú para ejecutar otras opciones de Pine. para hacerlo presione la tecla correspondiente a la letra que esta resaltada en el menú.

Ayuda

Para leer la ayuda en línea, no solo de la pantalla principal sino en de cualquier parte de PINE utilice el comando **?**. Para salir de la ayuda, teclear **E** (**Exit Help**).

Componer un mensaje

En el menú principal seleccione la opción Compose (C). Aparecerá la siguiente pantalla:



La parte superior se le conoce como cabecera del mensaje y la otra corresponde al "cuerpo" que es el mensaje propiamente dicho.

La cabecera contiene los siguientes campos:

- **To:** aquí se especifica la dirección a la que se va a enviar el mensaje. Si se desea enviar el mensaje a varias personas, escriba las direcciones de separadas por comas. Para agregar direcciones de la agenda presione <Ctrl><T>.
- **Cc:** en este campo se especifican las direcciones, separadas por comas, a las que se desee enviar una copia del mensaje. Si no desea enviar copias presione <enter>.
- **Attachmnt:** Este espacio se utiliza para incluir en el mensaje un archivo. Para ver la lista de archivos se pulsa <Ctrl><T>. Si no desea incluir ningún archivo en el mensaje, pulse <Enter>.
- **Subject:** Descripción del mensaje.

El cuerpo del mensaje inicia después de la línea "----- Message Text -----". Aquí es donde se escribe el mensaje propiamente dicho. Dentro de este parte se pueden ejecutar los siguientes comandos:

```
Ctrl-D Borrar el caracter en el que está el cursor
Ctrl-J Justificar el texto
Ctrl-Y Ir a la página anterior del texto
Ctrl-V Ir a la página siguiente del texto
Ctrl-K Borrar línea actual
Ctrl-U Recuperar la última línea que se ha borrado
```

Envío de un correo

Después de haber escrito su mensaje, para enviarlo presione <Ctrl><X> (<Send>). Confirme el envío del mensaje con <y> <enter>. Una copia del mensaje enviado se graba en el folder *sent-mail*.

Lista de mensajes

Para ver la lista de mensajes, debe presionar la tecla <I> en el Menú Principal, enseguida se presentará una pantalla con los mensajes recibidos. En la primera columna La pantalla tiene las siguientes columnas:

status del mensaje, que puede estar en blanco o contener alguno de los siguientes letras:

- N** si es un mensaje nuevo (que no se ha leído)
- +** si el mensaje ha sido enviado directamente
- A** si es un mensaje que se ha respondido
- D** si el mensaje ha sido seleccionado para ser borrado.

El resto de las columnas corresponde al número del mensaje, la fecha en que ha llegado el mensaje, la persona que lo envió, el tamaño y el asunto.

Leer un mensaje

Usando las flechas seleccione el mensaje y presione **V** (**ViewMsg**) o <enter>. Dentro del mensaje puede leer el mensaje siguiente presionando **N** (**NextMsg**) o volver a la lista de mensajes con la tecla **I** (**Index**)

Contestar un mensaje

Una vez seleccionado el mensaje que se desea responder en la pantalla **Folder Index**, escriba <R> (**Reply**), entonces el programa le preguntará si desea incluir el mensaje original en la respuesta:

```
Include original message in Reply?  
Y Yes  
^C Cancel N [No]
```

Entonces escriba el mensaje y proceda como en el caso de enviar un correo.

Reenvío de un correo (forward)

Si se desea volver a enviar un mensaje se utiliza el comando **F** (**Forward**) o el comando **B** (**Bounce**). La diferencia entre ambos es que **Bounce** sólo permite modificar la dirección a la que se va a enviar, mientras que **Forward** permite, modificar el resto de los campos de la cabecera y mensaje.

Borrar un mensaje

1. Para borrar un determinado mensaje, se le pone una "marca de borrado".
2. Utilice las flechas para seleccionar el mensaje que desea borrar.
3. **Presione D** (**Delete**) en cada uno de los mensajes en los que se desea marcar .
4. Cuando se salga de Pine, se pedirá que confirme si desea borrar definitivamente todos los mensajes que estén marcados.

```
Expunge the 8 deleted messages from "INBOX"?  
Y [Yes]  
N No
```

entonces:

si se presiona **y (yes)** se borrarán los mensajes marcados
si se presiona **n (no)** cuando se vuelva a entrar en el Pine los mensajes marcados
Para quitar una "*marca de borrado*" se utiliza el comando **U (Undelete)**.

Salida del programa Pine

- Para salir del programa, se presione la tecla **Q (Quit)** y el Pine nos pedirá confirmación:

```
Really quit pine?  
Y [Yes]  
N No
```

Comandos para comunicación interactiva

write

El comando write permite que un usuario le envíe mensajes a otros. Ambos deben estar conectados al mismo sistema. El usuario receptor debe tener habilitada la recepción de mensajes.

Para establecer la comunicación escriba el comando write seguido del user name del receptor.

```
$write maestro ↵
```

El receptor verá en pantalla el siguiente mensaje

```
Message from lety on Sun205 (pts/3) [ Sun Apr 20 00:21:35 ] ...
```

A partir de este momento en la pantalla del receptor aparecerá todo lo que el emisor escriba hasta que este presione <control><c>.↵

Si el usuario tiene deshabilitada la recepción de mensajes el mensaje no será enviado y se aparecerá en pantalla el siguiente mensaje:

```
$ write lety↵  
Permission denied.
```

Comando mesg

mesg: habilita/deshabilita los mensajes de otros usuarios enviados con write y la comunicación con talk.

- El comando mesg sin parámetros nos muestra si la recepción de mensajes está o no permitida

```
$ mesg  
is y
```

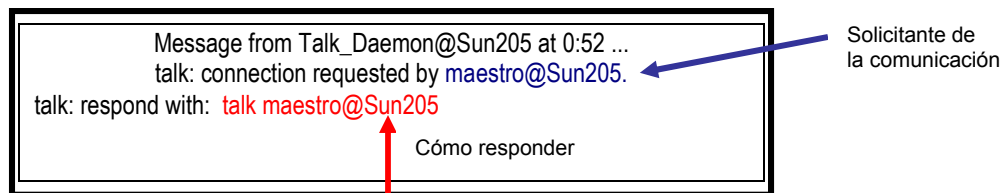
- Para deshabilitar la recepción de mensajes `$ mesg n`
- Para habilitar la recepción de mensajes `$ mesg y`

Comando talk

Los sistemas Unix permiten realizar charlas entre dos usuarios. Para establecer la sesión de **talk**, se requiere conocer el nombre de usuario del usuario con el que se desea comunicar y que éste se encuentre conectado al sistema.

Para establecer la comunicacion escriba el comando talk y enseguida el user name de su contraparte.

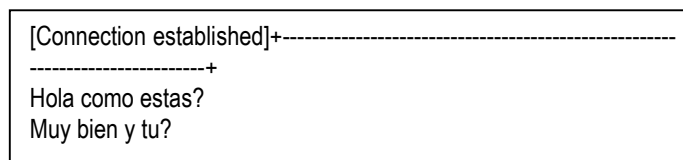
\$talk alumno←



En la pantalla del receptor se observará un mensaje como el siguiente:

Si el receptor desea establecer la comunicación debe escribir talk y el nombre del user name.

Para el ejemplo de la figura anterior bastara con escribir talk maestro. Una vez establecida la comunicación ambos usuarios se observa en la parte superior de la pantalla el mensaje [Coestablished]. A partir de este momento lo que escriban ambos usuarios se observara en la pantalla.



Para terminar la sesión de **talk**, se envía Ctrl-C en cualquiera de los dos extremos de la conexión.

FTP (File Transfer Protocol)

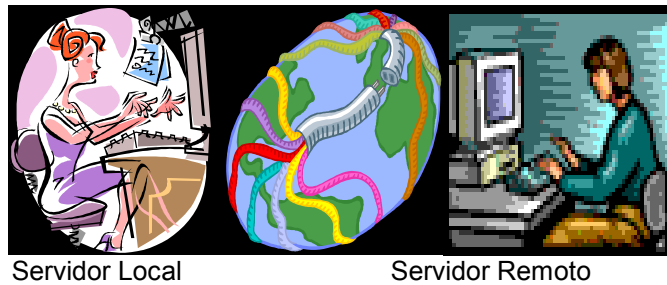
¿Qué es FTP?

FTP (File Transfer Protocol) Es un servicio que funciona con el protocolo TCP/IP. Constituye una de las herramientas más antiguas que utiliza Internet. Su uso es anterior a la creación de la World Wide Web. En sus inicios, FTP era utilizado por científicos e investigadores para transferir archivos de una computadora a otra. Luego, se crearon bibliotecas de archivos para que los usuarios pudieran tener acceso a los archivos sin necesidad de contraseñas. En resumen, FTP transfiere archivos desde y hacia una red remota.

Antes de establecer una conexión es necesario definir algunos conceptos.

El **servidor local** es la máquina donde se ejecuta ftp. Es desde nos conectamos para establecer la conexión.

El **servidor remoto** es el servidor al que nos conectamos para subir/bajar información.



Otro concepto importante que se debe manejar es la dirección del servidor, esta puede indicarse con la dirección IP o con el nombre de dominio.

Dirección IP: está formada por un cuatro bytes, entre 0 y 255, separados por puntos. Un ejemplo de una dirección IP válida puede ser 148.231.149.70 ó 148.231.149.200.

Nombre de Dominio. Este método consiste en asignar nombres a las computadoras, en el nombre, las palabras están separadas por puntos. Un nombre de dominio solamente puede estar relacionado con una IP. Por ejemplo, a la IP 148.231.149.200 le corresponde el nombre cq-ing.tij.uabc.mx.

Cuenta de acceso y contraseña: Para acceder a cualquier servidor remoto es necesaria una cuenta de acceso y password proporcionados por el administrador. Existen servidores públicos donde no es necesario identificarse, en este caso la cuenta de acceso se llama anonymous y como contraseña se ha generalizado usar nuestra dirección de correo electrónico, para que el usuario tenga referencia de quienes somos.

Establecimiento de la Conexión

Para establecer una sesión de FTP se necesita ejecutar el comando FTP ya sea desde la línea de comando o usando un navegador.

El uso de FTP en forma gráfica es sumamente sencillo. La pantalla se divide en forma vertical y en un extremo se observan los archivos de servidor local y en el otro los de servidor remoto. Para transferir archivos basta con marcarlo y arrastrarlo al lado opuesto. Entre los FTP con licencia gratuita podemos incluir [WS_FTP32](#), [LeechFTP](#), [Net Vampire 3.3](#), [SmartFTP 1.0 946](#) y [CuteFTP 4.2](#)

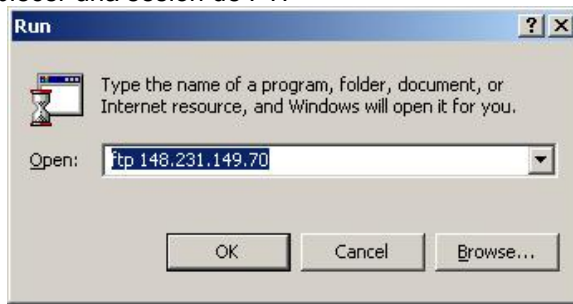
El FTP en línea de comando requiere del manejo de una serie de comando entre los más utilizados están los de la siguiente tabla:

Comando	Descripción
! <comando local>	Ejecuta un comando de la máquina local.
? <comando>	Muestra la ayuda del comando indicado.
append <archivo local><archivo remoto>	Agrega el archivo local al final del archivo remoto.
ascii	Establece que la transmisión será en formato ascii.
bell	Activa/Desactiva que la bocina suene al terminar un comando.
binary	Establece que la transmisión se hará en formato binario.
bye	Desconectar y terminar la sesión de ftp.
cd <directorio>	Cambiar al directorio remoto indicado.
delete <archivo>	Borra un archivo en el servidor remoto
dir	muestra el contenido del directorio remoto.
disconnect	terminar la conexión sin salir de ftp.
get <archivo>	Transferir un archivo del servidor remoto a la máquina local.
glob	Activa/Desactiva el uso de metacaracteres en el nombre de los archivos
help	Muestra la pantalla de ayuda
lcd	Cambia el directorio de trabajo local.
ls	Lista el directorio remoto.
mdelete	Borra múltiples archivos en el directorio remoto.
mget <archivo1> <archivo2> ...<archivoN>	Transferir varios archivos del servidor remoto a la máquina local. Se puede usar el metacaracter *.
mkdir <directorio remoto> <archivo local>	Genera un archivo en la máquina local con el contenido del directorio remoto.
mkdir	Crea un directorio en el servidor remoto
mput <archivo1> <archivo2> ...<archivoN>	Transfiere varios archivos de la máquina local al servidor remoto. Es posible el uso del metacaracter *.
open <IP o Nombre de servidor>	Establece una conexión con la dirección indicada
prompt	Activa/ Desactiva preguntar al usuario antes de cada operación.
put	Transfiere un archivo del servidor local al servidor remoto.
pwd	Muestra el directorio remoto actual
quit	Termina la sesión de FTP.
rename <nombreanterior> <nombre nuevo>	Renombrar un archivo en el servidor remoto
rmdir	Borrar un directorio en el servidor remoto
status	Muestra el estado actual de la conexión
type	Muestra el tipo de transferencia establecido.
user	Pide al usuario su user name y password

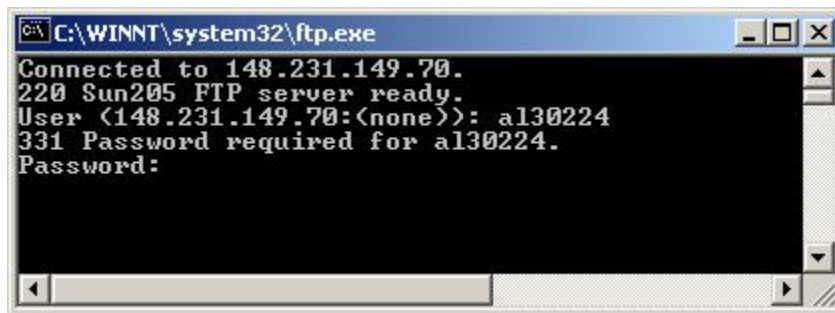
Transferencia de Archivos de la máquina remota al servidor. (Subir archivos)

Suponga que el usuario al30224 desea subir un archivo del directorio archivosJava de su computadora al directorio programas de su home directory en el servidor 148.231.149.70. Los pasos a seguir son los siguientes:

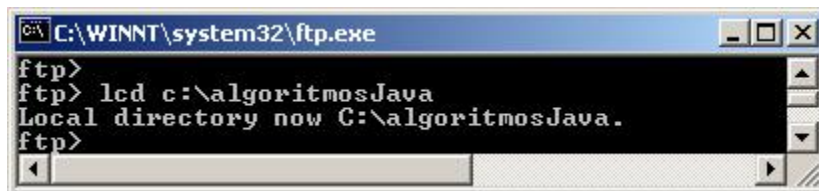
1) Establecer una sesión de FTP



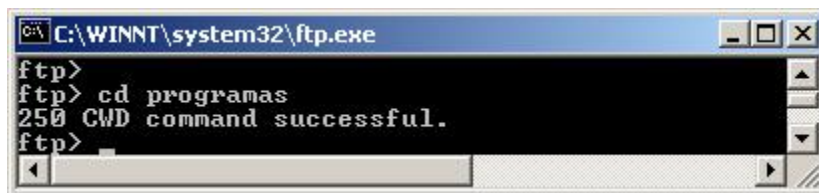
2) Introducir user name y password



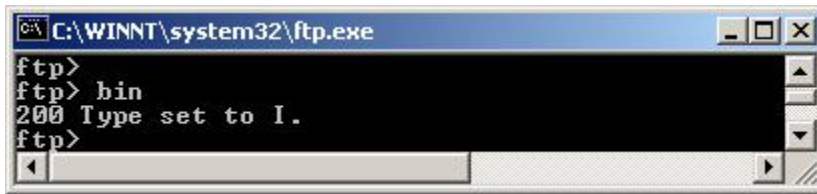
3) Establecer el directorio local; es decir, posicionarse en el directorio (en su pc) dónde están los archivos que se van a transmitir



4) Ubicarse en el directorio destino. Si el destino es su home directory este paso no es necesario.

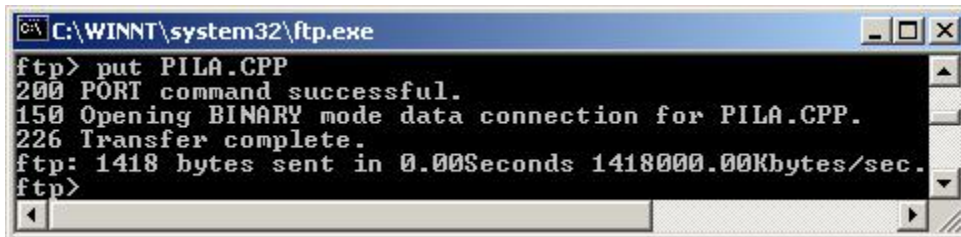


5) Indicar el formato de la transmisión.



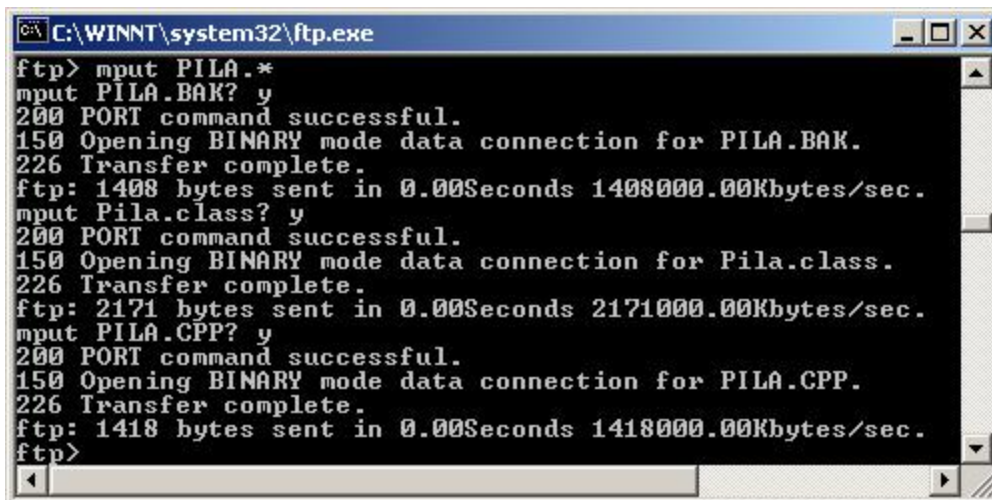
```
C:\WINNT\system32\ftp.exe
ftp>
ftp> bin
200 Type set to I.
ftp>
```

6) Para transferir el archivo use el comando put seguido del nombre del archivo.



```
C:\WINNT\system32\ftp.exe
ftp> put PILA.CPP
200 PORT command successful.
150 Opening BINARY mode data connection for PILA.CPP.
226 Transfer complete.
ftp: 1418 bytes sent in 0.00Seconds 1418000.00Kbytes/sec.
ftp>
```

Si va a transferir más de un archivo entonces se usa el comando mput y los comodines ? ó *
Por ejemplo si desea transferir todos los archivos llamados PILA la línea de comando sería la siguiente:



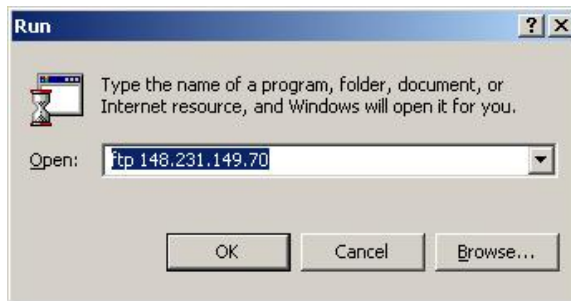
```
C:\WINNT\system32\ftp.exe
ftp> mput PILA.*
mput PILA.BAK? y
200 PORT command successful.
150 Opening BINARY mode data connection for PILA.BAK.
226 Transfer complete.
ftp: 1408 bytes sent in 0.00Seconds 1408000.00Kbytes/sec.
mput Pila.class? y
200 PORT command successful.
150 Opening BINARY mode data connection for Pila.class.
226 Transfer complete.
ftp: 2171 bytes sent in 0.00Seconds 2171000.00Kbytes/sec.
mput PILA.CPP? y
200 PORT command successful.
150 Opening BINARY mode data connection for PILA.CPP.
226 Transfer complete.
ftp: 1418 bytes sent in 0.00Seconds 1418000.00Kbytes/sec.
ftp>
```

En este caso se observa que está activado el modo interactivo; es decir, ftp pregunta al usuario antes de transferir cada uno de los archivos. El modo interactivo se desactiva/activa con el comando prompt.

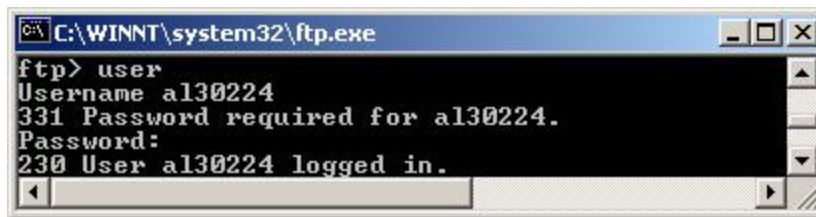
Transferencia de Archivos del servidor a la máquina remota. (Bajar archivos)

Suponga que el usuario al30224 desea transferir el archivo abril.txt del directorio reportes2004 de su home directory en el servidor 148.231.149.70 a un floppy. Los pasos a seguir son los siguientes:

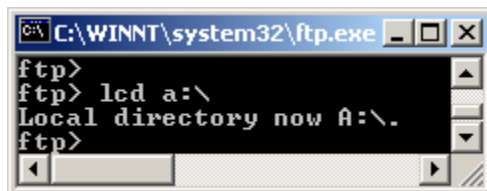
- 1) Establecer una sesión de FTP



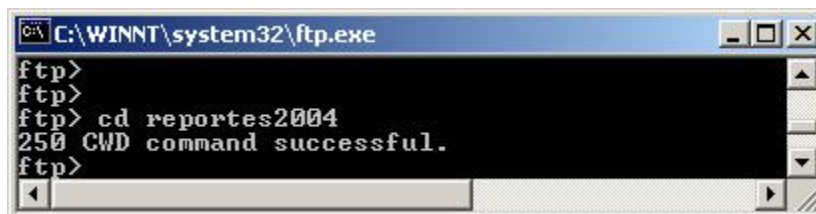
- 2) Introducir user name y password



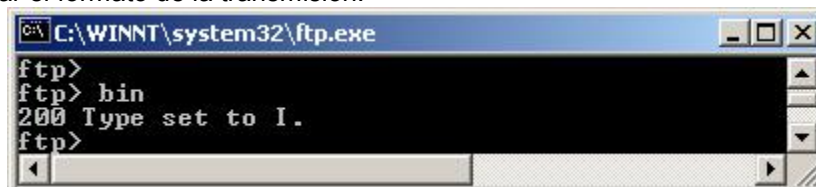
- 3) Establecer el directorio destino en la máquina local. En este caso el drive a.



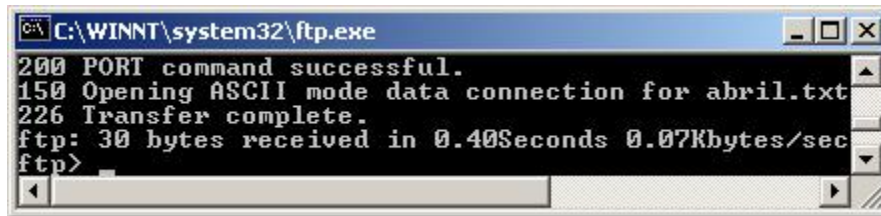
- 4) Posicionarse en el directorio fuente en la máquina remota.



- 5) Indicar el formato de la transmisión.

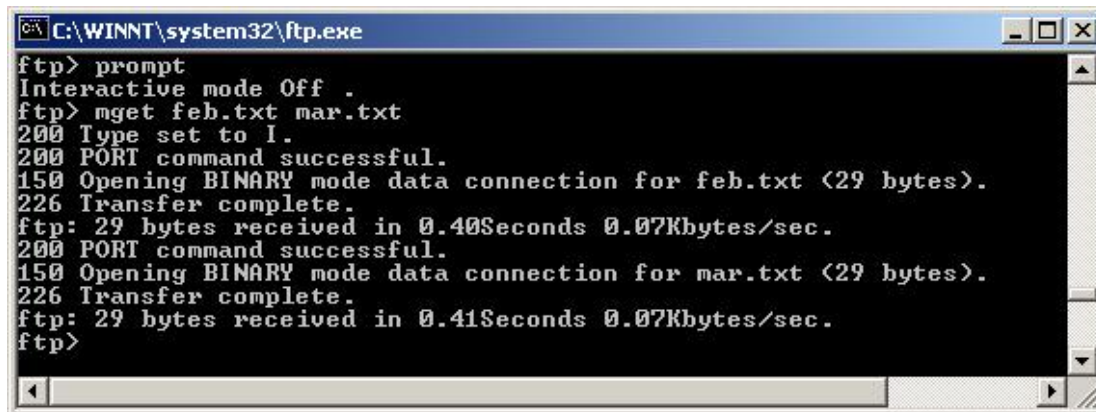


6) Para bajar el archivo abril.txt use el comando get



```
C:\WINNT\system32\ftp.exe
200 PORT command successful.
150 Opening ASCII mode data connection for abril.txt
226 Transfer complete.
ftp: 30 bytes received in 0.40Seconds 0.07Kbytes/sec
ftp>
```

Si desea transferir más de un archivo utilice mget con comodines, o los nombres de los archivos separados por espacios. Para desactivar el modo interactivo usaremos prompt.



```
C:\WINNT\system32\ftp.exe
ftp> prompt
Interactive mode Off .
ftp> mget feb.txt mar.txt
200 Type set to I.
200 PORT command successful.
150 Opening BINARY mode data connection for feb.txt (29 bytes).
226 Transfer complete.
ftp: 29 bytes received in 0.40Seconds 0.07Kbytes/sec.
200 PORT command successful.
150 Opening BINARY mode data connection for mar.txt (29 bytes).
226 Transfer complete.
ftp: 29 bytes received in 0.41Seconds 0.07Kbytes/sec.
ftp>
```

Desarrollo:

1. Envíele un correo al usuario maestro.
2. Encuentre a las personas reprobadas en el archivo lista. Envíeles un correo que contenga un mensaje (como attachment).
3. Encuentre a las personas aprobadas en el archivo lista. Envíeles un correo contenga un mensaje de felicitación (como attachment).
4. Usando el comando write envíele mensajes a tres de sus compañeros.
5. Usando el comando talk establezca comunicación con dos de sus compañeros.
6. Bloquee la recepción de mensajes.
7. Copie a su máquina el archivo pagina1.html que está en el directorio maestro/ftp/
8. Transfiera con una sola instrucción los archivos con extensión doc. del mismo directorio del punto anterior.
9. En Word escriba un párrafo sobre lo que hizo en sus vacaciones. Grabe el archivo.
10. Transfiera el archivo a su cuenta.
11. Renombre el archivo.
12. Transfiera el archivo con el nuevo nombre al disco duro o a un diskette.
13. Abra el archivo en word y verifique que el contenido sea el mismo que el original.
14. Verifique si tiene correo y si tiene contéstelo.
15. Transfiera tres o mas archivos (que tengan la misma extensión) en un solo paso.
16. Explique con sus propias palabras (en español) cinco comandos de FTP. Los comandos deberán ser diferentes a los vistos en clase.
17. ¿Qué es IP?
18. ¿Qué diferencia observa entre FTP y Telnet?
19. ¿Qué es DNS?

PRACTICA 7 EDITORES DE TEXTO

Introducción

Unix fue desarrollado en un ambiente en donde la terminal del usuario era un teletipo o algún otro tipo de terminal mecánica lenta; generalmente los monitores de video no se usaban. Un editor para ese ambiente era el orientado al trabajo en línea, en el cual el usuario ve y trabaja con sólo una línea de texto a la vez.

En sus inicios, UNIX se ponía a disposición de las universidades esencialmente de manera gratuita. Tanto los estudiantes como el personal docente de varias universidades contribuyeron al desarrollo del ambiente UNIX. De la Universidad de California en Berkeley surgieron mejoras notables. Una de ellas fue un editor de pantalla completa, en el cual se permite trabajar con toda una pantalla de información a la vez, en lugar de hacerlo con una sola línea. Este editor de pantalla completa se llama vi, que quiere decir **visual**. Desde la aparición de vi, han aparecido editores más potentes y fáciles de usar en muchos sistemas LINUX y UNIX. No obstante, siempre es conveniente conocer vi, ya que es el editor estándar en la mayoría de los sistemas UNIX y LINUX.

Acceso a vi. Para entrar a vi sólo necesita escribir vi en la línea de comando.

home\$ vi. Iniciar vi.

home\$ vi carta Iniciar vi abriendo el archivo carta. Si el archivo no existe lo crea.

Configuración de vi

vi -wn donde n es el número de líneas que tendrá el editor. Es útil cuando se trabaja en conexiones lentas. Ejemplo: \$vi -w10

Opciones del editor

:set number	Mostrar números de línea
:set nonu	No mostrar números de línea
:set autoindent	Activa la autoindentación

Al activarse vi la terminal se pone en blanco y aparece una tilde (~) en cada línea. En la parte inferior aparece el mensaje "empty buffer" si el archivo está vacío o es nuevo. Si el archivo no está vacío aparecerá una línea con el nombre del archivo, el total de líneas y el total de caracteres en el archivo. Al aparecer esta pantalla, se habrá arrancado vi correctamente; en este punto vi se encuentra en modo de comandos y en espera de la primera instrucción. Para pasar de modo de comandos a modo de edición, presione a para agregar texto o i para empezar a escribir su texto.

Modos de vi

Modo de escritura: El teclado funciona como una máquina de escribir. Para salir de este modo presiona la tecla <esc>.

Modo de comando: Las teclas del teclado son órdenes de edición.

Modo de edición de línea: Para entrar a este modo cambie a modo de comando y presione <shift> y <:>

Comandos para grabar el buffer de edición en un archivo

w	Graba el archivo que se está editando.
w <archivo>	Graba el archivo con el nombre que se indique.
w! <archivo>	Graba el archivo. Si el archivo ya existe lo sobrescribe.

Salida de vi

q	Salir del editor si el archivo ha sido grabado. Si no se ha grabado no podrá salir de vi.
q!	Salir del editor sin importar si el archivo ha sido grabado o no.
wq	Grabar el archivo y salir

Movimiento del Cursor

En muchos sistemas se pueden utilizar las flechas para mover el cursor, pero no en todos. Por esto, al desarrollarse vi se decidió utilizar las teclas h, l, k y j para mover el cursor.

h	Izquierda	
l	Derecha	
k	Arriba	
j	Abajo	

Otros movimientos del cursor.

w	Avanza una palabra.
b	Retrocede una palabra
\$	Desplazar el cursor hasta el final de la línea.
0 (cero)	Desplazar al inicio de la línea.
-	Desplazar al inicio de la línea anterior.
+	Desplazar al inicio de la línea siguiente.
ctrl-b	Pantalla anterior.
ctrl-d	Desplaza media pantalla adelante
ctrl-u	Desplaza media pantalla atrás
<n>G	Se desplaza a la línea n.

Comandos para escritura.

Para agregar texto al buffer de edición, se debe pasar del modo de comandos al modo de entrada. Cualquier caracter normal de texto que se escriba se agregará al buffer.

a	Agrega antes del cursor.
A	Agrega al final de la línea.
i	Insertar texto después del cursor.
I	Insertar texto al inicio de la línea.

De búsqueda

Los comandos de búsqueda abren una línea en la parte inferior de la pantalla donde se escribe el patrón a buscar. Dependiendo del comando que se escriba la búsqueda será hacia delante o atrás del cursor.

/	Busca un patrón a partir del cursor hacia adelante.
?	Busca un patrón del cursor hacia atrás.
N	Repite la búsqueda hacia delante.
n	Repite la búsqueda hacia atrás.

De borrado

Para eliminar texto, es necesario estar en el modo de comando. Todos los comandos eliminan caracteres a partir de la posición actual del cursor.

x	Borra un carácter
d	Borra el resto de la línea.
dd	Elimina una línea
<n>dd	Borra n líneas.
ctrl-r o ctrl-l	Limpia la pantalla de "basura" por errores de transmisión

De modificación

Para poder ejecutar estos comandos primero debe cambiarse al modo de comando. Los cambios se hacen en relación a la posición actual del cursor. Cada uno de estos comandos cambia al modo de edición, para capturar el texto que va a sustituir el anterior.

R	Reemplaza el caracter que está bajo el cursor. Para terminar la sustitución presione <esc>
c	Modifica una línea
C	
c	Cambia la palabra actual.
w	
r	Sobreescribir

Copiar, cortar e insertar texto.

Al eliminar o cortar caracteres, palabras o líneas, el objeto eliminado se guarda en el buffer para uso general. El objeto queda disponible para insertarse en cualquier parte del texto que está editando.

yw	Copia una palabra.
Yy	Copia la línea completa
<n>yy	Copia n líneas completas a partir de la actual.
p	Inserta texto copiado/borrado después del cursor.
P	Inserta texto copiado/borrado antes del cursor

Pico

El editor pico fue desarrollado en la Universidad de Washington. No obstante que forma parte del sistema de correo Pine, puede utilizarse de manera independiente. El término pico significa Pine Composer. Al iniciar pico queda en modo de escritura. Pico, a diferencia de vi tiene una interfaz amigable. El desplazamiento en pantalla y las funciones básicas se realizan a través de caracteres de control, los cuales vienen indicados en la parte inferior de la pantalla. El editor pico resulta atractivo por su facilidad de manejo, pero no es tan poderoso como vi, otra desventaja que tiene es que no está incluido en todos los sistemas UNIX ni LINUX como el editor vi.

Actividades

1. Copie el archivo amorosos.txt y encuentre el número de veces que se repite la palabra amorosos (Use el comando de búsqueda de vi para encontrar la palabra y lleve la cuenta manualmente)
2. Escriba al inicio el nombre del autor de los amorosos
3. Escriba al final su opinión acerca del poema.
4. Utilizando el comando cal y redireccionamiento directo genere un archivo que contenga el calendario del año en que nació.
5. Usando vi, edite el archivo, cambie el calendario al español (en forma manual) y marque el día de su nacimiento.
6. Modificar la información que se presenta con el comando finger.

PRACTICA 8. VARIABLES DE AMBIENTE

Variable

Es un espacio en memoria al cual se le da un nombre. Las variables son una forma de pasar información del shell a los programas al momento de ejecutarlos. Hay variables específicas que se crean al momento de entrar al sistema, pero también pueden haber variables que pueden ser definidas por el usuario.

Las variables en UNIX están divididas en dos categorías, **variables de ambiente y variables del shell**. Por convención, las variables de ambiente se nombran con mayúsculas y las variables del shell en minúsculas.

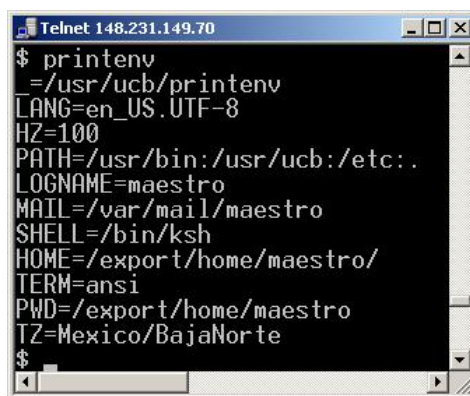
Variables de ambiente: Las variables de ambiente se usan para personalizar el entorno en el que se ejecutan los programas y para ejecutar en forma correcta los comandos del shell. Las variables de ambiente toman su valor inicial generalmente de un archivo `.profile`, pero hay veces en que el usuario tiene que modificar los valores de alguna variable de ambiente cuando está tratando de instalar o ejecutar un nuevo programa. Las variables de ambiente se usan durante toda la sesión de trabajo, ya que son heredadas por todos los procesos hijos.

Variables Built-in: Las variables Built-in son variables de ambiente que **no** pueden ser modificadas por el usuario. Sus valores son determinados por el sistema al inicio de la sesión. Las variables de este tipo más usuales son:

HOME	Contiene el home directory del usuario.
LOGNAME	Nombre de acceso al sistema del usuario.
USER	Nombre de acceso al sistema del usuario.
TZ	Huso horario usado por el sistema.

Variables shell: Las variables shell pueden contener cualquier valor. Solo están disponibles en el proceso en el que fueron creadas por lo que se usan para condiciones de trabajo a corto plazo.

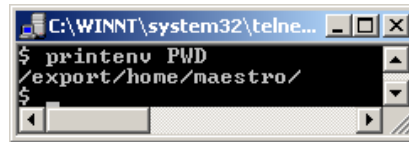
printenv: El comando `printenv` sin argumento, permite ver todas las variables de ambiente.



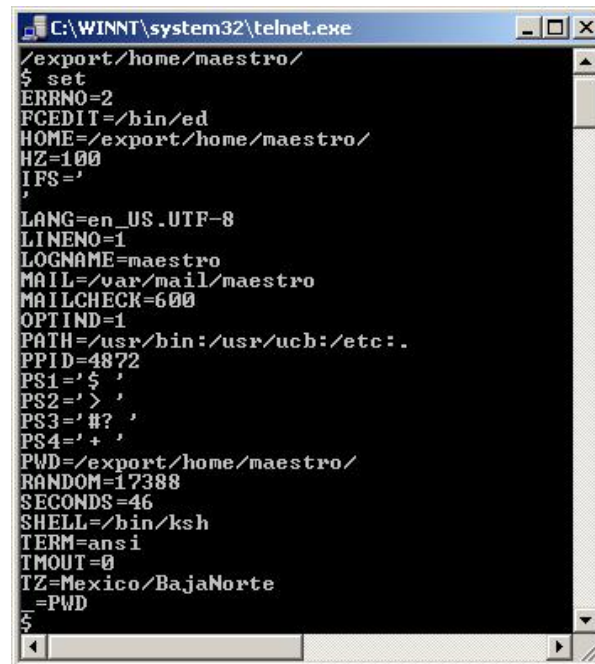
```
Telnet 148.231.149.70
$ printenv
=/usr/ucb/printenv
LANG=en_US.UTF-8
HZ=100
PATH=/usr/bin:/usr/ucb:/etc:.
LOGNAME=maestro
MAIL=/var/mail/maestro
SHELL=/bin/ksh
HOME=/export/home/maestro/
TERM=ansi
PWD=/export/home/maestro
TZ=Mexico/BajaNorte
$
```

En el ejemplo anterior el sistema muestra las variables de ambiente. Cada sistema puede listar más o menos dependiendo de cómo lo haya configurado el administrador. Las variables de ambiente se definen en los archivos de arranque o inicio de sesión cuando un usuario entra al sistema.

Para ver el valor de una variable específica se usa el mismo comando seguido de la variable.



El comando **set** muestra una lista de todas las variables de ambiente y de shell con sus respectivos valores actuales.



Creación de variables

Para crear una variable de shell, simplemente se asigna un valor a la variable. La sintaxis es la siguiente:

variable=valor.

Ejemplo: nueva=1.

Asignación de valores a variables de ambiente:

Muchas de estas variables especiales son automáticamente definidas por el sistema y les asigna un valor inicial al entrar a la sesión. El usuario puede modificarlas de la siguiente manera:

Variable=nuevovalor.

Exportación de variables del shell al ambiente: Para pasar una variable ya existente la sintaxis es **export Variable.**

Si la variable no ha sido creada puede hacerse de las siguientes formas:

a) **variable=valor.**

export variable
ó en una sola línea

b) **export variable=valor.**

Eliminación de variables

Para eliminar una variable de shell se escribe unset y el nombre de la variable.

```
$unset nueva.↵
```

En el shell csh una variable de ambiente se elimina con unsetenv y el nombre de la variable.

```
Sun205% unsetenv nueva.↵
```

Lista de Variables de ambiente en UNIX.

IFS	Contiene una lista de caracteres que se van a emplear como caracteres internos de separación de campo
LANG	Lenguaje
LINENO	Contiene el número de línea de un script que está siendo ejecutado actualmente.
LOGNAME	Nombre con el que se abrió la sesión de trabajo
MAIL	Contiene el nombre de ruta del inbox.
MAILCHECK	Frecuencia con la que se revisa el correo. Por default 600 segundos.
OPTIND	Contiene el siguiente argumento de línea que será procesado.
PATH	La variable PATH contiene una lista separada por dos puntos (:) de los directorios donde el shell busca los comandos para su ejecución.
PPID	Número de proceso del padre del proceso shell.
PS1	Indicador primario, se usa cuando el shell es interactivo. Por default es \$.
PS2	prompt string 2. Se usa cuando es necesario completar la entrada. Por default es <
PS3	prompt string 3. Contiene la cadena de petición usada en combinación con la palabra reservada select. Por default es #?.
PS4	prompt string 4. Contiene el prefijo para comandos trazados con set -x. Valor default es +.
PWD	Directorio de trabajo actual.
RANDOM	Asigna el valor de la semilla para el generador de números aleatorios.
SECONDS	Contiene el tiempo transcurrido desde el inicio de la sesión. (en segundos).
SHELL	Nombre y ruta del shell actual que el usuario está usando
TERM	Tipo de terminal en el que se estableció la conexión para presentar correctamente la información en la terminal.
TMOUT	Tiempo de espera para que el usuario ingrese su user name.
TZ	time zone

Actividades:

1. Muestre todas las variables que hay en su sesión de trabajo.
2. Muestre solamente las variables de ambiente.
3. Modifique el valor de una de las variables de ambiente.
4. ¿Se puede modificar nuestro user name? Si/No ¿por qué?
5. ¿Cual es la diferencia entre las líneas de comando \$echo PATH y \$echo \$PATH?
6. Cree una variable de shell con su nombre y asignele su matricula como valor.
7. Cree una variable de ambiente con su nombre y asignele su matricula como valor.
8. Modifique el prompt primario
9. Modifique el prompt secundario.
10. Elimine la variable de ambiente que creó en el punto 7.
11. Investigúe que otras variables built-in existen en unix. Anexe su investigación al reporte. No cuenta como información adicional.

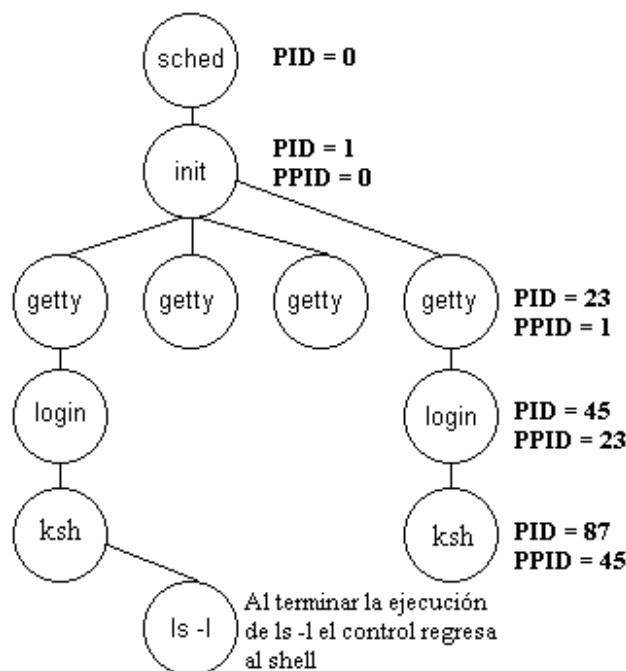
PRÁCTICA 9 PROCESOS

Definición de Proceso

Proceso es un concepto fundamental para todo sistema operativo. Es una entidad dinámica que consiste en un programa en ejecución, sus valores actuales, su estado y los recursos utilizados para manejar su ejecución (memoria, CPU, dispositivos de E/S, etc). Pueden coexistir varias instancias de un mismo programa en ejecución en forma simultánea. ya que cada una de ellas es un proceso diferente.

Programa: Un programa es una entidad inactiva, estática, la cual consiste de un conjunto de instrucciones y datos asociados. Si un programa es invocado varias veces puede generar múltiples procesos.

Unix es un sistema multiproceso por tiempo compartido. Aunque en cualquier momento muchos procesos parecen estar ejecutándose concurrentemente desde el punto de vista del proceso, este tiene el acceso y control de todos los recursos del sistema como si fuera único. La realidad es que la mayoría de los sistemas Unix corren en plataformas que tienen una unidad de procesamiento capaz de soportar muchos procesos activos; sin embargo, en un instante dado de tiempo solamente un proceso puede estar en ejecución. Los sistemas operativos Unix cambian rápidamente (en milisegundos) el proceso actual en ejecución, aparentando así que trabajan en forma concurrente; es decir, aparentan que trabajan en varios procesos al mismo tiempo pero en realidad, existe una distribución temporal de la asignación de CPU entre los diferentes procesos que compiten por ejecutarse. La conmutación temporal de procesos, está basada en un sistema de colas FIFO multinivel con actualización de prioridad. Los sistemas con múltiples unidades de procesamiento, las cuales por definición pueden soportar un verdadero procesamiento concurrente se dice que tienen capacidad de multiprocesamiento. En la figura puede ver que los procesos tienen una estructura jerárquica en árbol:



Tipos de procesos:

- **Procesos del sistema:** Son los procesos que actúan sin que el usuario los solicite. También se les conoce como daemons. Pueden ser de dos tipos:
 - 1) Procesos permanentes o de larga duración: Se crean al arrancar el sistema y permanecen activos hasta que se termina la conexión. Su función es realizar las actividades del sistema.
 - 2) Procesos transitorios: Nacen y mueren cuando el sistema efectúa tareas propias, independientes de los usuarios.
- **Procesos de usuario:** Son los procesos creados cuando el usuario ejecuta comandos.

Atributos del proceso

- Cada proceso tiene un identificador único (PID). Este es un entero no negativo asignado por el sistema. Garantiza que el proceso sea único dentro del sistema.
- Un proceso tiene asignados tres segmentos de memoria: Segmento de código, Segmento de datos (datos estáticos) y Segmento de Stack (datos dinámicos).
- Identificador del usuario y grupo al que pertenece
- Identificadores de otros procesos relacionados con él
- Datos de los sucesos que le harán despertarse y señales pendientes.
- Tamaño del proceso
- Datos de su planificación.
- Terminal Original.

Prioridades de los Procesos

Una manera de minimizar los tiempos de respuesta del procesador para aprovechar de forma eficiente los recursos del sistema es que mientras un usuario realiza sus procesos otros usuarios pueden tener tareas en ejecución de manera simultánea.

- Dejar trabajos en baja prioridad en el sistema es una buena manera de utilizar los recursos eficientemente.
- Los procesos interactivos (los que requieren entrada/salida) necesitan mayor prioridad.
- El sistema ajusta las prioridades de los procesos dinámicamente, de tal forma que se puede ajustar a cambios en los procesos.
- Los procesos que no utilizan el procesador por un periodo aumentan su prioridad.

El comando **nice** permite dejar un proceso en baja prioridad. Dependiendo de la versión de UNIX acepta un rango diferente de prioridades. Pero los números más altos tienen prioridad más baja.

nice < -incremento /decremento> <comando>

Ejemplo: \$ nice -1vi

Si no se usa nice la prioridad será establecida por el sistema. Solamente root puede incrementar la prioridad de un proceso.

Estados del proceso

Durante la ejecución de un proceso su estado va cambiando, pudiendo encontrar en cualquiera de los siguientes estados:

- Estado Activo en Ejecución (Running): El proceso tiene asignado un tiempo de CPU y las instrucciones se están ejecutando.
- Estado Activo Ejecutable (Runnable): El proceso puede ser ejecutado, pero no tiene tiempo de CPU disponible.
- Estado Suspendido: Recibió una señal para detenerse (SIGSTOP). Continuará cuando reciba SIGCONT.
- Durmiendo: Se encuentra en espera de un evento, por ejemplo una entrada de teclado, o que otros procesos terminen.
- Ocioso(Idle): Fue creado pero aún no es ejecutable.
- Zombi: El proceso terminó pero su padre no ha sido notificado.

El comando ps reporta el estado de los procesos activos. `$ ps`

Si no se especifican opciones, ps mostrará información sobre los procesos que tienen los mismos user ID y controlador de terminal que la conexión desde donde se llama a ps. La información que muestra ps dependerá de las opciones especificadas, por omisión contiene el número de proceso, identificador de terminal, tiempo de ejecución acumulado y el nombre del comando que se está ejecutando.

Modalidades de ps

-a	Muestra los procesos de otros usuarios.
-c	Muestra el nombre del comando.
-e	Lista información sobre cada proceso en ejecución ahora.
-f	Muestra mas información sobre los procesos. Incluyendo el nombre del usuario.
-l	Genera una lista con información detallada de los procesos.
-u	user visualiza los procesos de dicho usuario.
-j:	La información se presenta empezando por el PID.
-p proclist.	Lista solamente los procesos cuyos números de identificación están en la lista.
-t term.	Lista los procesos asociados con la terminal. Por ejemplo, term/a, o pts/0.
-u uidlist	Lista los procesos del user ID o login name especificado(s). La lista debe estar separada por comas.
-o formato	Muestra la información de acuerdo a un formato especificado en una lista separada por comas. El formato se especifica usando una o más de las siguientes palabras: user ruser group rgroup uid ruid gid rgid pid ppid pgid sid taskid pri opri pcpu pmem vsz rss osz nice class time etime stime f s c lwp nlwp psr tty addr wchan fname comm args projid project pset.

Ejemplos: `$ ps -u root`
`$ps -u root -o pid, pri, nice, time`

Cada columna de la información que se visualiza con ps describe el estado del proceso.

S	O Corriendo
	S En espera de un evento sea completado.
	I Ocioso.
	Z Zombie
	T Parado por una señal del padre.
UID	El número ID del usuario.
PID	Número del proceso.
PPID	Número del proceso padre.
PRI	Prioridad del proceso.

ADDR	Dirección de memoria del proceso.
TIME	Tiempo acumulativo de ejecución.
CMD	Nombre del comando. (Nombre completo con -f)
NI	El valor de prioridad proceso.
SZ	El tamaño virtual de la imagen, calculado como el tamaño de texto+pila+datos
RSS	El tamaño del conjunto residente. El número de Kilobytes del programa que está residente en la memoria actualmente
WCHAN	El número del evento del kernel por el que está esperando el proceso
TTy	El nombre del terminal de control para el proceso

Ejecución en background y foreground

Desde la línea de comando los procesos pueden crearse de dos maneras:

- **Primer Plano (foreground):** El shell espera a que el proceso termine de ejecutarse para volver a mostrar el prompt. Ejemplo: **\$cat>tarea**
- **Segundo Plano (background):** El proceso se ejecuta con una prioridad menor, el shell no espera a que termine y mientras tanto el usuario puede ejecutar otro comando. Para mandar una tarea (job) al background el comando se escribe finalizando con un "&". El shell notifica el PID asignado al proceso. Ejemplo:

```
$vi& ↵  
[1] 8925
```

Donde [1] es el número de tarea y 8925 el número de PID del proceso.

Jobs

A un programa corriendo en el segundo plano o background se le conoce como tarea. El comando jobs muestra una lista de todos los procesos que un usuario esta ejecutando como tarea.

Ejemplo:

```
$jobs↵  
[1] + Stopped (SIGTTOU) vi&
```

La información anterior indica que la tarea1 (vi) fue detenida

\$fg↵ Pasa una tarea del background al primer plano. Se puede usar especificando el pid o el número de tarea.

a) Con el pid \$fg <pid> Ejemplo: **\$fg 8925↵**

b) Con el número de tarea \$fg %<no.tarea1> Ejemplo: **\$fg %1↵**

Eliminación de procesos

El comando kill se usa para enviar señales a los procesos. Algunas de estas señales son las siguientes:

kill -HUP <pid> : Señala al proceso con numero <pid>, que vuelva a leer sus archivos de configuración

kill -INT <pid> : Señala al proceso con numero <pid>, que sera interrumpido

kill -TERM <pid> : Señala al proceso con numero <pid>, que debe de terminar, a diferencia de -KILL , esta opción da la oportunidad al proceso de terminar.

kill -STOP <pid> : Señala al proceso con numero <pid>, que pare momentaneamente

kill -CONT <pid> : Señala al proceso con numero <pid>, que continúe, este comando se utiliza para reanudar un proceso que le fue aplicado -STOP

kill -KILL <pid> : Señala al proceso con número <pid/notarea>, que termine de inmediato.

Las señales tiene un número asignado en la tabla de señales, en el caso de KILL es (9). La sintaxis de kill es: **kill <señal> <pid/no.tarea>**

Ejemplo: \$kill -9 8995↵

Para eliminar una tarea **\$kill -9 %1↵**

El resultado de los ejemplos anteriores es el mismo si se escribe **\$kill -KILL 8995↵**

Comando nohup

Al terminar una sesión de Unix, el sistema mata todos los procesos del usuario. Si algún proceso requiere mayor tiempo de ejecución el comando nohup permite que un proceso continúe ejecutándose aunque el usuario haya terminado su sesión.

Sintaxis:

nohup comando &

Ejemplo: **\$nohup find / archivo&**

Señales (signals)

Las señales en Unix son la forma para avisar a un proceso que ha sucedido cierto evento y que debe ser atendido. La lista de señales a comunicar a los procesos varía muy poco entre las versiones de Unix.

Nombre	Valor	Acción	Evento
SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status Changed
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Exit	Pollable Event
SIGIO	22	Exit	input/output possible signal
SIGSTOP	23	Stop	Stopped(signal)
SIGTSTP	24	Stop	Stopped(user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped(tty input)
SIGTTOU	27	Stop	Stopped(tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGCKPT	33	Ignore	Checkpoint warning
SIGRESTART	34	Ignore	Restart warning
SIGRTMIN	49	Exit	POSIX1003.1b SIGRTMIN
SIGRTMAX	64	Exit	POSIX1003.1b SIGRTMAX

Tabla de Señales del sistema operativo UNIX IRIX

Desarrollo:

1. Genere un listado completo de todos los procesos que están en el sistema y muestre la información completa de todos los que se empezaron a ejecutar el 12 de mayo (en una sola línea).
2. Qué están haciendo los procesos que actualmente esta ejecutando maestro. (Comando)
3. Genere un listado con el número de proceso, número del proceso padre, comando en ejecución y prioridad de tres de sus compañeros.
4. Explique la diferencia entre las opciones de ps e,f,l y j
5. Explique la diferencia entre las opciones de ps a y u
6. Explique qué es lo que hace la opción de ps t y u
7. Si tiene dos sesiones de telnet abiertas con el mismo user name
8. ¿qué procesos muestra al ejecutar ps?
9. ¿Qué opción de ps debería de usar para ver todos los procesos de un usuario?
10. ¿Cómo identifico a los procesos que el usuario está ejecutando en cada terminal?
11. ¿Cuál es significado de TODAS las columnas de formato que maneja ps -o? (Sólo las que no están explicadas en este material).
12. Ejecute dos comandos en background (los que quiera).
13. Ejecute el comando cat >lista, ¿Qué prioridad tiene asignada?
14. Mate el proceso anterior.
15. Vuelva a ejecutar cat>lista pero con menor prioridad.
16. ¿Qué prioridad le fue asignada?
17. Una vez más ejecute cat>lista, pero ahora en el background .
18. ¿Cuál es su prioridad ahora?
19. Verifique que el comando en background este en la lista de procesos.
20. Verifique que el comando en background este en la lista de tareas (jobs).
21. Pase una de las tareas al foreground (use el número de tarea)
22. Pase la otra tarea al foreground, pero ahora use el número de **PID**.
23. Envíe otro comando al background.
24. Finalice este proceso.
25. Compruebe el funcionamiento del comando nohup.

PRÁCTICA 10. ALIAS Y SCRIPTS

Comando alias

En ocasiones se suelen utilizar comandos que son difíciles de teclear o de recordar. UNIX ofrece la posibilidad de dar un alias o nombre corto a un comando con o sin parámetros, para que cada vez que se quiera ejecutar, sólo se escriba el alias.

Definición de un alias.

La sintaxis es **alias <nombre_alias>="<comando>"**

Ejemplo: **\$ alias lista="ls -l"** ↵

Nota: Si el alias ya existía se sobrescribe.

Comando unalias

Para borrar un alias creado dentro de nuestra sesión de trabajo, se utiliza el comando unalias. La sintaxis es **unalias <nombre_alias>**

Ejemplo: **\$ unalias lista** ↵

Programación del Shell

Shell Script. Un shell script es un archivo de texto ordinario, contiene una colección de uno o más comandos del shell.

Ventajas del uso de shell scripts

- Evita escribir secuencia de comandos que son largas o complicadas.
- Permite automatizar tareas.
- Evita errores.

Pasos para la creación de un shell script

- Generar el archivo de texto con un editor como vi, pico, emacs, etc.
- Modificar los permisos para hacer el archivo ejecutable.
- Ejecute el script con **\$ksh <shell_script>**

Reglas para escribir un shell script

- Use sangrías para separar las líneas del script, así será más fácil de leer.
- El enter se considera el fin de comando. Para continuar un comando en la línea siguiente escriba una diagonal inversa (\) al final de la línea.
- Los espacios, tabuladores y líneas en blanco son irrelevantes.
- Para agregar comentarios a sus scripts simplemente escriba **#** al inicio de la línea.

Comandos del shell

clear. Limpiar pantalla

echo. Sirve para enviar datos a la salida estándar, estos datos pueden ser cadenas o valores de variables. Automáticamente agrega el salto de línea, a menos que se especifique la opción -n.

echo "Hola"	Envía a la pantalla la palabra Hola.
echo \$saludo	Muestra en pantalla el contenido de la variable saludo.
echo "Hola /c"	"Hola" Envía el mensaje a la pantalla, pero no hace salto de línea.

read. Este comando lee el valor para las variables desde el teclado. El delimitador es el espacio. Si el usuario escribe más de una palabra se asigna a la primera variable y el resto a la segunda. Generalmente se usa acompañado de echo ya que es necesario enviar un mensaje (echo) para que indique al usuario que se necesita un dato de entrada.

read variable	Lee una variable
read variable1 variable2	Lee dos variables

ejemplos:

```
echo "Como te llamas?"  
read nombre
```

```
echo "Cual es tu nombre y matrícula?"  
read nombre matricula
```

Asignación directa de valores a variables

Para darle un valor a una variable simplemente se escribe el nombre de la variable (sin espacio), seguido del signo de igual y enseguida el valor. La sintaxis es la siguiente:
nombre-variable=valor

Nota: Cuando el valor de la variable es una cadena de caracteres debe ir entrecomillado.

Ejemplos: domicilio= "Calzada Tecnológico"
 edad=26

Variables Posicionales (argumentos de la línea de comando)

El shell emplea variables posicionales para almacenar los argumentos o parámetros del script. La variable 0 es el nombre del programa, la 1 corresponde al primer argumento, la 2 al segundo y así sucesivamente. Considere la línea \$ agenda amigo 5674.

La variable posicional 0 contiene el nombre del script: agenda
La variable posicional 1 contiene el primer parámetro escrito: amigo
La variable posicional 2 contiene el segundo parámetro escrito: 5674

Acceso a las variables posicionales.

```
echo "El script en ejecucion es $0"
echo "Hola $1"
```

Ejercicio: Paso 1) Teclee en vi las siguientes líneas. Grábe con el nombre ej7

```
# Script ej7. Ejemplo de variables posicionales
ps -u $1 -o pid,ppid, nice
```

Paso 2) Otorgue permisos de ejecución

Paso 3) Ejecute la línea **\$ej7 maestro**

Variables Predeterminadas

- \$# :num. de argumentos
- \$* :todos los arg de la shell
- \$\$:PID de la Shell
- \$HOME
- \$PATH

Sustitución de comando

La comilla invertida indica al shell que la cadena entrecomillada es un comando a ejecutarse, por ejemplo, en el siguiente script la variable fecha toma el valor que resulta de ejecutar el comando date. Esto es lo que se conoce como sustitución de comando.

Ejemplo de sustitución de comando.

```
clear
actual=`pwd`
echo " El directorio actual es $actual"
```

Caracteres especiales dentro de los scripts

Los siguientes símbolos son considerados caracteres especiales dentro de los scripts para poderlos usarlos dentro de una cadena (en el comando echo) es necesario anteponerles la diagonal invertida o backslash [\].

\$	Hace referencia al contenido de una variable.
	Entubamiento.
#	Comentario
&	Ejecución de un proceso en el background.
?	Comodín para un solo caracter.
*	Comodín para varios caracteres.
>	Redireccionamiento de salida.
<	Redireccionamiento de entrada.
`	Sustitución de comando.
>>	Redireccionamiento de salida.
[]	Rango de caracteres.
[a-z]	Caracteres de a a z.

Actividades

1. Cree un alias para los comandos `date`, `cp` y `finger`.
2. Hacer un shell script que muestre solamente la hora del sistema.
3. Hacer un shell script que muestre el directorio actual y el nombre real del usuario.
4. Hacer un shell script que pida el nombre de un archivo, busque el archivo y muestre su contenido, el número de líneas, palabras y caracteres que contiene.
5. Hacer un shell script que pida los nombres tres archivos, sustituya los caracteres de mayúsculas a minúsculas en el primero, cambie todos los números por asteriscos al segundo y el tercero permanece igual. Muestre en pantalla los tres archivos y pida al usuario el nombre del archivo donde se guardarán los tres archivos.
6. Hacer un shell que pida el nombre real de una persona, encuentre su nombre de usuario, muestre los procesos que está ejecutando y le envíe un mensaje en tiempo real.
7. Elimine el alias para el comando `cp`.
8. Hacer un shell que tome de la línea de comando un número `N` y un nombre de archivo. Muestre las primeras y últimas `N` líneas del archivo especificado en la línea de comando.
9. Hacer un shell que lea el nombre de un archivo, enseguida pregunte los permisos que sobre el archivo, desea otorgar para usuario, grupo y otros. (Los permisos se pedirán en forma individual para cada entidad)
10. Hacer un shell que lea las medidas de un terreno rectangular y determine el costo que tendrá cercar el terreno si considera que el precio por metro lineal de material es de 150 pesos y la mano de obra tiene un costo de 100 pesos. Se pide mostrar el perímetro del terreno, el total por concepto de mano de obra, el total por concepto de material y el costo total de la construcción.

REFERENCIAS

1. El libro de Unix
Sarvar, Koretsky y Sarwar
Editorial Addisson Wesley
2. Aprendiendo UNIX
James Gardner
Prentice Hall
3. Manual de Referencia de HP UX
4. Manual de Referencia de Solaris