EBook Gratis

APRENDIZAJE unix

Free unaffiliated eBook created from
Stack Overflow contributors.

#unix

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: unix

It is an unofficial and free unix ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official unix.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con Unix

Observaciones

Esta sección proporciona una descripción general de qué es Unix y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de Unix y vincular a los temas relacionados. Dado que la Documentación para Unix es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar Unix.

Lea Empezando con Unix en línea: https://riptutorial.com/es/unix/topic/912/empezando-con-unix

Capítulo 2: Comandos básicos de la consola

Examples

pwd - imprimir directorio de trabajo

```
$> pwd
/home/myUserHome
$> cd ..
$> pwd
/home
```

imprimirá la ruta actual a la consola.

pushd / popd (almacene el directorio actual en la pila y vaya a dest / pop prev dir y vaya a él)

```
$ pwd
/home/bob/somedir1/somedir2/somedir3

$ pushd /home/bob/otherdir1/otherdir2
/home/bob/otherdir1/otherdir2 /home/bob/somedir1/somedir2/somedir3

$ popd
/home/bob/somedir1/somedir2/somedir3

$ pushd /usr
/usr /home/bob/somedir1/somedir2/somedir3

$ pushd /var
/var /usr /home/bob/somedir1/somedir2/somedir3

$ popd
/usr /home/bob/somedir1/somedir2/somedir3

$ pwd
/usr

$ popd
/home/bob/somedir1/somedir2/somedir3

$ pwd
/home/bob/somedir1/somedir2/somedir3
```

comandos de manipulación de archivos

Lista de comandos que serán introducidos aquí:

```
ls     #view contents of a directory
touch  #create new file
mkdir  #create new directory
cp     #copy contents of one file to another
```

```
mv     #move file from one location to another
rm     #delete a file or directory
```

ls ejemplos

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     test.cpp
```

muestra el directorio actual

```
jennifer@my_computer:~/Desktop$ ls c++\ projects
DNA_analysis.cpp        encryption.cpp  pool_game.cpp
```

Muestra el directorio "c ++ proyectos". Los caracteres de espacio en los nombres de archivo se escriben como "\".

ejemplo táctil

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     test.cpp
jennifer@my_computer:~/Desktop$ touch ruby_test.rb
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby_test.rb    test.cpp
```

ejemplo mkdir

```
jennifer@my_computer:~/Desktop$ mkdir ruby
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby    ruby_test.rb    test.cpp
jennifer@my_computer:~/Desktop$ cd ruby
jennifer@my_computer:~/Desktop/ruby$ ls
<nothing>
jennifer@my_computer:~/Desktop/ruby
```

En realidad no se imprime `<nothing>` . Es solo que estoy representando que no produce nada.

ejemplos de cp

```
jennifer@my_computer:~/Desktop/ruby$ cd ..
jennifer@my_computer:~/Desktop$ cp test.cpp c++_test.cpp
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby_test.rb
test.cpp
```

Esto es cuando el último argumento a `cp` , en este caso "c ++ _ test.cpp" no es un directorio existente. `cp` creará un archivo llamado "c ++ _ test.cpp", con contenido idéntico al de "test.cpp". Si c ++ _ test.cpp ya existiera, `cp` habría eliminado lo que había antes antes de copiar el contenido de "test.cpp".

```
jennifer@my_comptuer:~/Desktop$ ls ruby
<nothing>
```

```
jennifer@my_computer:~/Desktop$ cp ruby_test.rb ruby
jennifer@my_computer:~/Desktop$ ls ruby
ruby_test.rb
```

Esto es lo que sucede cuando el último argumento de `cp` , en este caso "ruby", es un directorio. `cp` crea un archivo con el mismo nombre que "ruby_test.rb", pero en el directorio "ruby".

mv ejemplos

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby_test.rb
test.cpp
jennifer@my_computer:~/Desktop$ mv ruby_test.rb ruby\ test.rb
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
test.cpp
```

Esto es lo que sucede cuando el último argumento a `mv` , en este caso "ruby test.rb", no es un directorio existente. El archivo "ruby_test.rb" ha sido renombrado a "ruby test.rb". Si "ruby test.rb" ya existiera, se habría sobrescrito. Note, nuevamente, que los espacios están precedidos por un ".

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
test.cpp
jennifer@my_computer:~/Desktop$ ls c++\ projects
DNA_analysis.cpp        encryption.cpp  pool_game.cpp
jennifer@my_computer:~/Desktop$ mv test.cpp c++\ projects
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
jennifer@my_computer:~/Desktop$ ls c++\ projects
DNA_analysis.cpp        encryption.cpp  pool_game.cpp   test.cpp
```

Esto es lo que sucede cuando `mv` es un directorio que ya existía. El archivo "test.cpp" se mueve al directorio "proyectos de c ++".

rm ejemplos

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
jennifer@my_computer:~/Desktop$ rm c++_test.cpp
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby    ruby test.rb
```

c ++ _ test.cpp ha sido eliminado

```
jennifer@my_computer:~/Desktop$ rm c++\ projects
rm: cannot remove 'c++ projects': Is a directory
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby    ruby test.rb
```

`rm` tiene un requisito adicional para eliminar directorios

```
jennifer@my_computer:~/Desktop$ rm -rf c++\ projects
jennifer@my_computer:~/Desktop$ ls
Research Paper.docx     ruby    ruby test.rb
```

`-rf` debe agregar para eliminar un directorio.

Para obtener más información sobre `ls` , escriba el comando `ls --help` . Para `touch` , escriba `touch --help` . Del mismo modo con los 6 comandos mencionados aquí. Esto imprime una explicación detallada de uso sin crear o eliminar nada.

comando cd, directorios explicados

```
michael@who-cares:~$
```

El símbolo `~` después de `who-cares:` es el directorio actual. `~` realidad significa el directorio personal de la persona. En este caso, eso es `/home/michael` .

```
michael@who-cares:~$ cd Downloads
michael@who-cares:~/Downloads$
```

Busca `Downloads` en el directorio actual, luego hace que el directorio actual.

```
michael@who-cares:~/Downlaods$ cd /var
michael@who-cares:/var$
```

Dado que este directorio comenzó con un `/` , eso significa buscar en el directorio raíz el directorio `var` . Para aquellos que vienen de Windows, el directorio raíz es el equivalente a `C:\` . Los directorios que comienzan con `/` se llaman **"directorios absolutos"** y los directorios que no se llaman **"directorios relativos"**

```
michael@who-cares:/var cd lib/dbus
michael@who-cares:/var/lib/dbus$
```

El `/` en el medio significa hacer `cd lib` y una vez que se hace `cd dbus` en un comando.

```
michael@who-cares:/var/lib/dbus$ cd .
michael@who-cares:/var/lib/dbus$
```

`.` en realidad significa "el directorio actual". El comando `cd .` es básicamente inútil, pero `.` Es útil para otras cosas.

```
michael@who-cares:/var/lib/dbus$ cd ..
michael@who-cares:/var/lib$
```

`..` realidad significa "el padre del directorio actual". Como tal, `cd ..` significa "navegar por un directorio hacia arriba".

```
michael@who-cares:/var/lib$ cd ../log/apt
```

```
michael@who-cares:/var/log/apt$
```

`.` y `..` también puede ser parte de la `/` cadena. Además, no hay límite de cuánto tiempo puede ser.

```
michael@who-cares:/var/log/apt$ cd /dev/bus
michael@who-cares:/dev/bus$
```

La cadena `/` incluso puede existir cuando el directorio comienza en la raíz.

```
michael@who-cares:/dev/bus$ cd /
michael@who-cares:/$
```

`cd /` te lleva al directorio raíz. Me pregunto qué pasará si escribes `cd ..` aquí ... (no te preocupes, es seguro)

```
michael@who-cares:/$ cd home
michael@who-cares:/home$ cd michael
michael@who-cares:~$
```

Cada usuario tiene un directorio para sus cosas dentro del directorio de inicio. Si el directorio actual está en el directorio de inicio, esa parte del nombre, en este caso `/home/michael` , se reemplazará por `~` .

```
michael@who-cares:~$ cd sys
michael@who-cares:/sys$ cd ~/Desktop
michael@who-cares:~/Desktop$ cd ~/..
michael@who-cares:/home$
```

`~` También puede ser parte de la `/` cadena. Incluso puede estar en la misma cadena que `..` Si el directorio comienza con `~` , es un directorio absoluto como si comenzara con `/` .

Lo último que debe intentar: escriba `cd` sin directorio después.

cual

Para determinar en qué parte de su sistema existe un ejecutable en su ruta, use el comando `which` :

```
$ which python
$
```

Si no hay respuesta, ese ejecutable no existe en su ruta. El sistema simplemente le devolverá un nuevo mensaje sin un mensaje de error. Si el ejecutable existe en su ruta, mostrará el directorio donde realmente existe:

```
$ which ls
/bin/ls
```

Esto puede ser útil para determinar por qué el comportamiento no coincide con las expectativas al garantizar que está ejecutando la versión del ejecutable que cree que es. Por ejemplo, si tiene ambos Python 2 y Python 3 instalados, ambos podrían ejecutarse escribiendo `python` en el terminal, pero el ejecutable que se está ejecutando puede ser diferente de lo esperado. Como se muestra arriba, este comando funcionará para cualquier comando estándar de Unix, todos respaldados por ejecutables individuales.

Comandos básicos de Unix

```
$pwd
```

Muestra el directorio de trabajo actual.

```
$who
```

Muestra todos los usuarios conectados.

```
$who am i
```

Muestra el nombre de usuario del usuario actual.

```
$date
```

Muestra la fecha actual del sistema.

```
$which <command>
```

Muestra la ruta del comando especificado. Por ejemplo, "$ which pwd" mostrará la ruta del comando 'pwd'.

```
$file <file_name>
```

Muestra el tipo del archivo especificado (archivo normal, directorio u otros archivos)

```
$cal
```

Muestra el calendario del mes actual.

```
$bc
```

Muestra el cálculo matemático entre dos enteros de flotadores. Por ejemplo, "$ bc 2 + 3" devolverá la suma aritmética de 3 y 5.

```
$ls
```

Enumera los contenidos del directorio.

- $ ls -l: listas en formato largo.
- $ ls -c: salida de múltiples columnas.
- $ ls -f: Muestra el tipo de archivo.
- $ ls -r: listado recursivo de todos los subdirectorios encontrados.
- $ ls -a: muestra todos los archivos, incluidos los archivos ocultos.
- $ ls -i: muestra todos los archivos junto con su número de I-Node.

```
$grep [options] <pattern> <input_file_names>
```

Imprime líneas que contienen una coincidencia para un patrón.

- $ grep -i: Realiza emparejamiento insensible a mayúsculas
- $ grep -v: imprime todas las líneas que no contienen la expresión regular
- $ grep -r: Busca recursivamente los subdirectorios listados e imprime los nombres de los archivos con la aparición del patrón
- $ grep -I: Excluye archivos binarios

Lea Comandos básicos de la consola en línea:
https://riptutorial.com/es/unix/topic/4262/comandos-basicos-de-la-consola

Capítulo 3: Comenzando con los comandos de Unix

Introducción

Este tema proporcionará una cobertura completa de los comandos básicos de Unix.

Examples

Una lista no exhaustiva de comandos de Unix

```
$man <command>
```

Muestra las páginas de manual en línea para el comando

```
$clear
```

Borra la pantalla del terminal

```
$pwd
```

Devuelve el nombre del directorio de trabajo

```
$echo <string>
```

Escribe la cadena en la salida estándar

```
$printf <string>
```

Formatee e imprima la cadena Ejemplo: imprima $ PATH $ printf “% s \ n” $ PATH

```
$uptime
```

Mostrar cuánto tiempo ha estado funcionando el sistema

```
$which <program>
```

Localiza un archivo de programa en la ruta del usuario.

```
$whereis <program>
```

Comprueba los directorios binarios estándar para los programas especificados, imprimiendo el

ARCHIVOS

```
$cd [directory]
```

Cambiar directorio Símbolos de directorio comúnmente utilizados:

- . : Directorio actual
- .. : Directorio de padres
- ~: Directorio de inicio
- / : Directorio raíz

```
$ls
```

- $ ls -a: Mostrar archivos ocultos
- $ ls -l: Mostrar lista larga
- $ ls -1: Mostrar solo el nombre del archivo por línea
- $ ls -h: formato legible por humanos

  $ archivo Determine el tipo de archivo (por ejemplo, gzip)

Archivos de lectura

```
$more
```

Muestra el contenido de un archivo una pantalla a la vez.
barra espaciadora: desplácese a la siguiente pantalla; b = pantalla anterior
entrar: desplazarse una línea
h: Ayuda para más
q: dejar de ayudar

```
$less <file>
```

Menos es un programa similar a más, pero que permite el retroceso en el archivo así como el avance.

```
$cat <file>
```

Lee archivos de forma secuencial, escribiéndolos en la salida estándar

```
$head [-number] <file>
```

Mostrar las primeras líneas de un archivo

```
$tail [-number] <file>
```

Muestra el contenido del archivo o, por defecto, su entrada estándar, a la salida estándar

- $ tail -f: Ver cambios en el archivo en tiempo real

```
$touch <file>
```

Establece la modificación y tiempos de acceso de los archivos. Si algún archivo no existe, se crea con los permisos predeterminados.

```
$tee <file>
```

Copia la entrada estándar a la salida estándar. Presione ctrl-d para dejar de agregar contenido

- $ tee -a: agregue la salida a los archivos en lugar de sobrescribirlos

```
$mkdir <directory>
```

Crear un directorio

```
$wc
```

Muestra el número de líneas, palabras y bytes contenidos en cada archivo de entrada o entrada estándar

- wc -l: contar líneas
- wc -w: contar palabras
- wc -m: contar personajes

```
$diff <file1> <file2>
```

Compara dos archivos línea por línea. Imprimirá sólo las diferentes líneas.

```
$locate <file>
```

Localiza los archivos en el disco

- $ localizar -q: suprimir errores

```
$find <path> <expression> <action>
```

Buscar archivos por nombre o contenido

- $ find -name: Buscar por nombre de archivo
- $ find -size <+/- n> Ejemplo: Encuentre archivos en el directorio actual que sean más grandes que 10k $ find. -size +10

```
$rm <file or directory>
```

Eliminar <archivo o directorio>

- rm -f: Saltar confirmación
- rm -i: aprobar cada eliminación
- rm -r: recursivo

Ejemplo: Eliminar el y es contenido $ rm -r

```
$mv <source_file> <target_file>
```

Renombra un archivo

```
$mv <source_file> <target_directory>
```

Mueve un archivo

- $ mv -i: no anular archivos de salida
- $ mv -r: recursivo

Ejemplo: mover directorio arriba en jerarquía $ mv ..

```
$cp
```

Copie un archivo / directorio dentro de la misma máquina (use el comando scp para copiar a una máquina remota)

- $ cp -i: no anular archivos de salida
- $ cp -r: recursivo

Ejemplo: copiar y renombrar un archivo

```
$cp <file_name> <new_file_name>
```

Ejemplo: copiar al directorio

```
$cp <file_name> <directory_name>
```

Ejemplo: copiar y renombrar un directorio

```
$cp –R <directory> <new_directory>
```

Ejemplo: copiar todos los archivos de tipo específico a un directorio

```
$cp *.txt <directory>
```

```
$ln –s <file> <link name>
```

Crear un alias (enlace) a un archivo

- $ ln -s: crea un enlace flexible (un enlace que funciona a través de máquinas)

```
$sort <file>
```

Ordene el contenido de un archivo -r ordenes inversas -n ordenación numérica

Ejemplo: ordene y escriba el resultado en sorted.txt $ sort | uniq -u> ordenado.txt

```
$uniq [-ucd] filename(s)
```

Busca líneas duplicadas. Los datos deben ser ordenados primero

- $ uniq -d: muestra solo una copia de las líneas duplicadas
- $ uniq -u: Mostrar solo líneas que no están duplicadas
- $ uniq -c: Genere cada línea precedida por un conteo de ocurrencias Ejemplo: muestre a los usuarios que están conectados más de una vez $ who | corte -d '' -f1 | ordenar uniq -d

```
$grep <pattern> <file_name>
```

Imprime líneas que contienen una coincidencia para un patrón.

- $ grep -i: Realiza emparejamiento insensible a mayúsculas
- $ grep -v: imprime todas las líneas que no contienen la expresión regular
- $ grep -r: Busca recursivamente los subdirectorios listados e imprime los nombres de los archivos con la aparición del patrón
- $ grep -I: Excluye archivos binarios

```
$tr “string1” [“string 2”]
```

Herramienta de búsqueda y reemplazo. Tr solo acepta su entrada desde tuberías y redirecciones. no acepta archivos como entrada.

- tr -d: borra todas las apariciones de todos los personajes en string1

Ejemplo: Imprima a.txt en la pantalla después de eliminar todas las apariciones de ";" $ cat a.txt | tr -d “;”

- tr -s: Reemplaza las ocurrencias con un solo carácter

Ejemplo: $ echo “SSSS SS” | tr -s “S” “S”

```
$tar
```

Crea y manipula archivos de transmisión de archivos. Esta implementación puede extraer imágenes tar, pax, cpio, zip, jar, ar e ISO y puede crear archivos tar, pax, cpio, ar y shar.

USO DEL DISCO

```
$du [file or directory]
```

Muestra el uso del bloque del sistema de archivos para cada archivo o directorio. Si no se especifica ningún archivo / directorio, se muestra el uso del bloque del directorio actual.

- $ du -a: Archivos y directorios (el valor predeterminado es solo directorios)
- $ du -h: Formato legible por humanos Ejemplo: Mostrar el uso del disco, ordenado, solo archivos MB $ du -h | grep -i “m \ t” | ordenar -n Ejemplo: Averigüe los 10 principales archivos / directorios más grandes $ du -a / var | sort -n -r | cabeza -n 10

```
$df
```

Mostrar espacio libre en disco

- $ df -h: formato legible por humanos

REDIRECCIONES Y TUBOS

> Redirigir la salida estándar. No sobrescriba el archivo si existe

>! Redirigir la salida estándar. Sobrescribir el archivo si existe

> & Redirigir la salida estándar y el error estándar

Ejemplo: redirigir la salida del comando a un archivo

```
$ls > result.txt
```

Utilice el archivo> / dev / null para eliminar el mensaje de error

Ejemplo: busque un archivo llamado my_file_name e imprima el resultado en ~ / find.txt; Ocultar errores (por ejemplo, "permisos denegados")

```
$find / -name my_file_name.* > /dev/null > ~/find.txt
```

<Redirigir la entrada estándar

>> Agregar salida estándar <comando> >>: Agregar salida al final de una existente

<comando> <: redirigir la entrada a un comando desde un archivo

| Redirigir la salida estándar a otro comando (pipe)

Ejemplo: Mostrar detalles paginados de procesos en ejecución

```
$ps –ex | more
```

| : La salida de la tubería del comando 1 es la entrada de la orden2 (si se desea un archivo de salida en el medio de una tubería, use el comando tee)

Ejemplo: contar el número de usuarios conectados

```
$who | wc –l
```

PROCESOS

```
$ps
```

Mostrar procesos activos

- $ ps -e: Mostrar información sobre el proceso
- $ ps -x: Mostrar procesos ocultos

Ejemplo: Encuentre procesos por nombre $ grep -l <process_name_regex>

```
$kill [–signal] pid
```

Matar un proceso.

Algunas de las señales más utilizadas:

- 3: SALIR (salir)
- 9: KILL (kill no capturable, no ignorable)
- 15: PLAZO (señal de terminación de software)

```
$top
```

Visualice y actualice la información ordenada sobre los procesos. Consulte las páginas del manual para obtener una lista de posibles claves. Las claves comunes son: CPU, hilos, puertos.

- $ top -o

```
$htop
```

Muestra y actualiza la información ordenada sobre los procesos.

PERMISOS DE USUARIO

```
$sudo <command>
```

Ejecutar el comando como superusuario.

```
$su
```

(usuario sustituto) abre una sesión como administrador

```
$exit
```

Salir de la raíz

```
$whoami
```

Mostrar ID de usuario efectivo

```
$who
```

Imprimir todos los nombres de usuario conectados

```
$passwd
```

Cambia la contraseña

```
$chmod <who> <operation> <permissions> <file or directory name>
```

Cambiar el acceso de propietario / grupo a un archivo o directorio

Quién: u usuario; grupo g; o otro; todo lo anterior

Operación: + añadir; - retirar; = establecer (lo que significa restablecer a nada y establecer solo lo que se especificó)

Permisos: rwx

Ejemplo: Agrega permisos de lectura / ejecución al grupo

```
$chmod g +rx <file>
```

Ejemplo:

```
$chmod 743 <file>
```

Tenga en cuenta que para $ cd en un directorio necesita los permisos x

Lea Comenzando con los comandos de Unix en línea:
https://riptutorial.com/es/unix/topic/9848/comenzando-con-los-comandos-de-unix

Capítulo 4: Descripción general de Unix

Introducción

Desarrollado en AT&T Bell Labs por Ken Thomson como sistema operativo de un solo usuario en 1969. Inicialmente escrito en lenguaje ensamblador Desarrollado como sistema operativo multiusuario. más tarde reescrito en C en 1973 Licenciado para la universidad con fines educativos en 1974 POSIX (Sistema operativo portátil para Unix) fue desarrollado

Examples

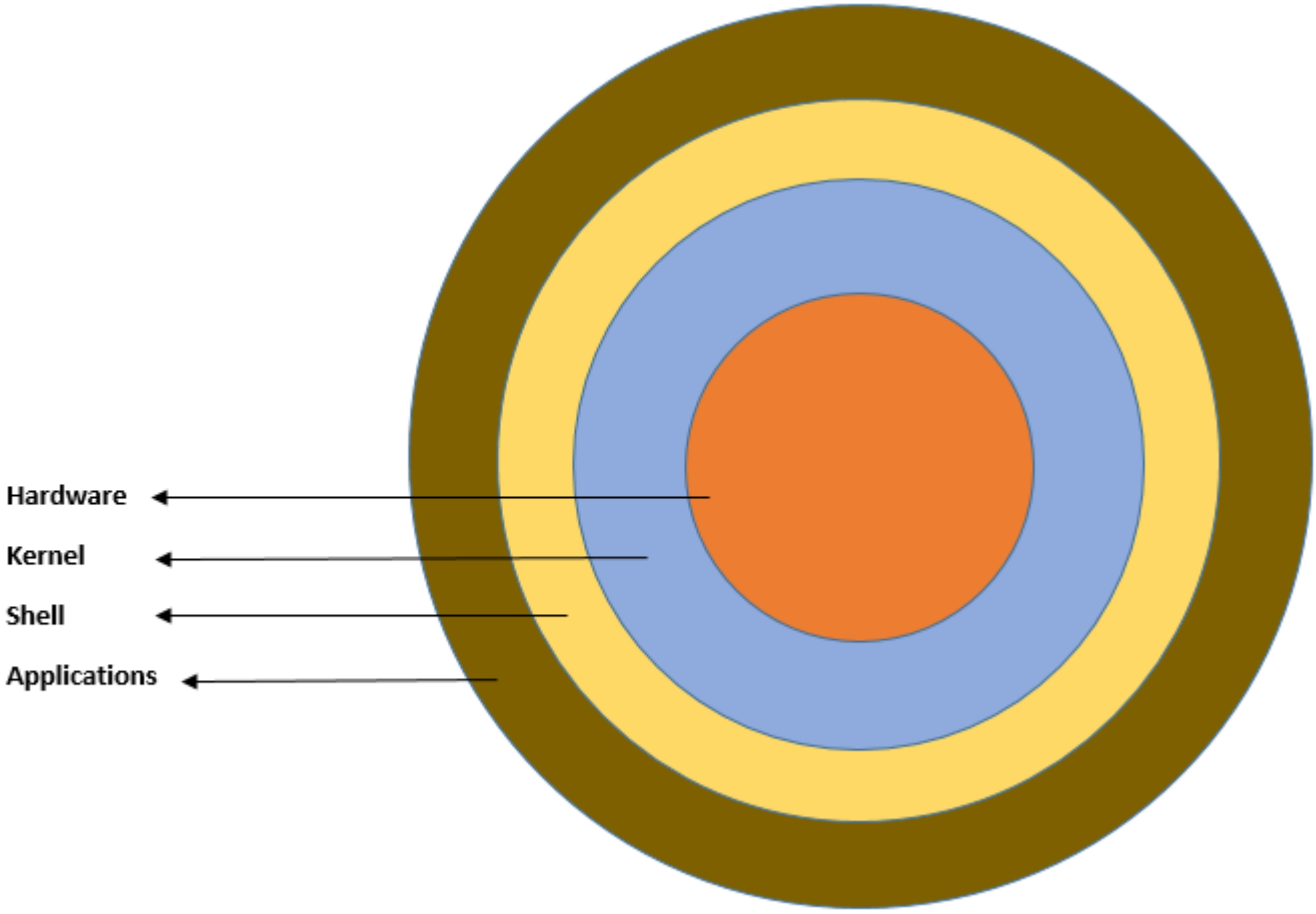
Sabores de Unix

- AIX por IBM
- Solaris de Sun Microsystems
- HP-UX por Hewlett Packard
- IRIX de Silicon Graphics, Inc
- FreeBSD de Free BSD Group
- GNU / Linux por Open Source Movement
- SCO Unix por The Santa Cruz Operation Inc

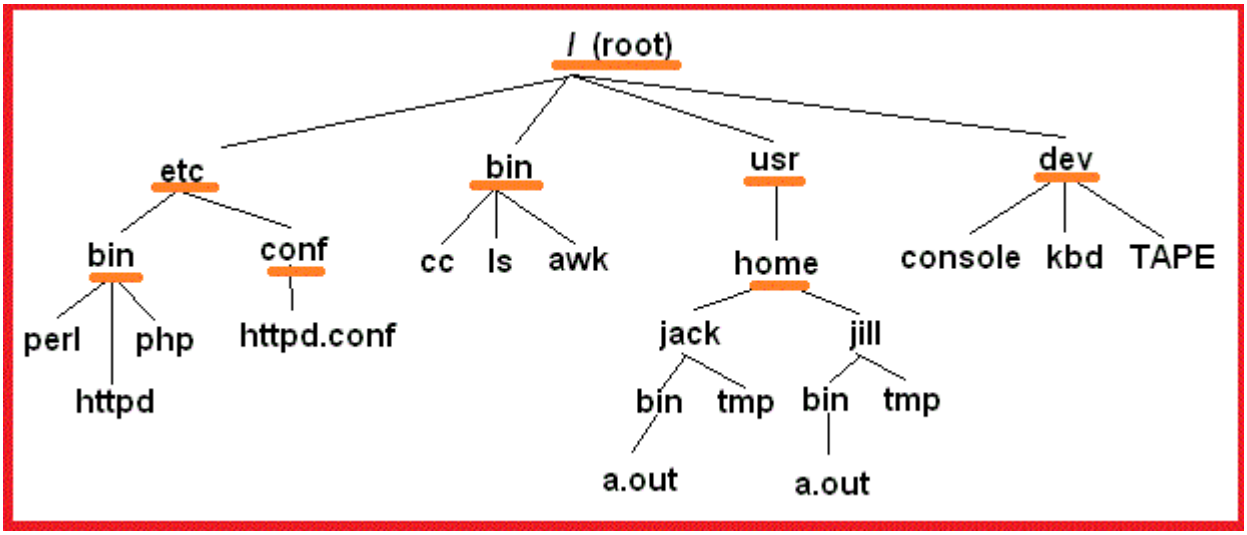
Características de UNIX

- Multi usuario
- Multitarea
- Interactivo
- Cáscara
- Seguridad
- Sistema de archivos jerárquico

Arquitectura de Unix



Sistemas de archivos Unix



Lea Descripción general de Unix en línea: https://riptutorial.com/es/unix/topic/9338/descripcion-general-de-unix

Capítulo 5: Permisos

Introducción

En Unix cada archivo tiene ciertos permisos como leer, escribir y ejecutar. Un usuario puede manipular los permisos de un archivo usando el comando 'chmod'.

Observaciones

En UNIX, hay tres permisos utilizados para otorgar un cierto nivel de acceso a un archivo o carpeta.

Para archivos:

- **Leer** : Permite al usuario / grupo / otros leer un archivo.
- **Escribir** : Permitir que el usuario / grupo / otros modifique un archivo.
- **Ejecutar** : permite al usuario / grupo / otros ejecutar (o ejecutar) un archivo.

Estos son ligeramente modificados para directorios:

- **Leer** : Permite al usuario / grupo / otros listar los nombres de los archivos en un directorio.
- **Escribir** : permite al usuario / grupo / otros crear, eliminar y renombrar archivos en un directorio.
- **Ejecutar** : permite al usuario / grupo / otros acceder a los metadatos y contenidos de un directorio.

Estos permisos se pueden representar utilizando las letras "r" para leer, "w" para escribir y "x" para ejecutar. También se pueden representar numéricamente: 4 para lectura, 2 para escritura y 1 para ejecución.

Examples

Cambiar los permisos de un archivo

```
> chmod 644 example.txt
> ls -l example.txt
-rw-r--r--  1 owner ogroup 57 Jul  3 10:13  example.txt
```

El comando anterior cambia los permisos del archivo para permitir que el propietario del archivo lea y escriba en un archivo. También permite que los usuarios del grupo del propietario y otros usuarios del sistema lean el archivo.

Entendiendo permisos

Digamos que hay un archivo que nos gustaría ejecutar, un script de bash llamado `add.sh` , por

ejemplo. `./add.sh` embargo, al escribir `./add.sh` , `./add.sh` un error de permiso. Obtener los permisos es un proceso simple.

Para determinar los permisos que tiene un archivo, escriba:

`ls -l filename` , o, en nuestro caso, `ls -l ./add.sh`

Esto imprime lo siguiente en la consola:

```
-r--r--r-- 1 username groupname 0 Jan 4 12:00 add.sh
```

Paremos y entendamos lo que esto significa. Hay tres tipos diferentes de permisos: propietario, grupo, otros. Permisos distintos se aplican a cada tipo de permiso.

También hay tres acciones de permisos, que en términos más generales también describen lo que un usuario puede hacer exactamente con un archivo. Estos son: (lee: r, escribe: w, ejecuta: x).

Entonces, regresa a esa serie de guiones y r's. Cada grupo de permisos tiene tres habilidades potenciales. Los grupos se enumeran en el orden `owner-group-others` y las acciones como `read-write-execute` .

Pero espera, eso significa que hay un carácter adicional al principio de la cadena. Este es en realidad el caracter descriptor del archivo. Podemos ver que hay un `-` existe, pero existen otros caracteres para cosas como directorios `(d)` , sockets `(s)` , enlace simbólico `(l)` etc.

Esto nos deja esencialmente con esta información: `a file where owner, group, and others have read permissions. No other permissions granted.`

Vamos a modificar esto para permitir que el propietario también escriba y ejecute el archivo. Nota: Dependiendo de los permisos, puede ser necesario anteponer `sudo` a este comando.

```
chmod 744 add.sh
ls -l add.sh
```

Imprime

```
-rwxr--r-- 1 username groupname 0 Month time add.sh
```

Ahora, el propietario del archivo puede ejecutar el archivo escribiendo

```
./add.sh
```

Cálculo CHMOD

Cálculo CHMOD

CHMOD es binario. _ / _ _ _ / _ _ _ / _ _ = _ / 4 + 2 + 1/4 + 2 + 1/4 + 2 + 1 = 777 = _ / rwx / rwx / rwx = 777 Por lo tanto _rwx = _ / 4 + 2 + 1 = 7

D / _ _ _ / _ _ _ / _ _ _ ('D' = directorio, otro uso es L = Enlace)

Así, por ejemplo, _rwxr_xr_x = _ / rwx / rx / r_x = 755

Abajo

Para cambiar el propio: grupo usas el comando chown usuario: grupo

por ejemplo, chown owner: group o si el propietario y el grupo son iguales, puede usar chown owner: (porque linus asume que owner: group es el mismo).

Lea Permisos en línea: https://riptutorial.com/es/unix/topic/6394/permisos

Capítulo 6: Ver las páginas de manual

Examples

Viendo la página del manual para un comando del sistema

```
man <command>
```

Esto mostrará la página de manual para el comando especificado.

Por ejemplo, `man ping` mostrará:

```
PING(8)                   BSD System Manager's Manual                  PING(8)

NAME
     ping -- send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS
     ping [-AaCDdfnoQqRrv] [-b boundif] [-c count] [-G sweepmaxsize]
          [-g sweepminsize] [-h sweepincrsize] [-i wait] [-k trafficclass]
          [-l preload] [-M mask | time] [-m ttl] [-P policy] [-p pattern]
          [-S src_addr] [-s packetsize] [-t timeout] [-W waittime] [-z tos]
          host
     ping [-AaDdfLnoQqRrv] [-b boundif] [-c count] [-I iface] [-i wait]
          [-k trafficclass] [-l preload] [-M mask | time] [-m ttl] [-P policy]
          [-p pattern] [-S src_addr] [-s packetsize] [-T ttl] [-t timeout]
          [-W waittime] [-z tos] mcast-group

DESCRIPTION
     The ping utility uses the ICMP protocol's mandatory ECHO_REQUEST datagram
     to elicit an ICMP ECHO_RESPONSE from a host or gateway.  ECHO_REQUEST
     datagrams (``pings'') have an IP and ICMP header, followed by a ``struct
     timeval'' and then an arbitrary number of ``pad'' bytes used to fill out
     the packet.  The options are as follows:

...
```

Mientras se visualiza la página del manual se puede buscar. Escribir una barra (`/`) seguida por el término de búsqueda saltará a la primera aparición del término. Ejemplo: `/ping`

Si presiona `N` después, saltará a la siguiente aparición. `Shift+N` saltará a la ocurrencia anterior.

Obtener la ruta de archivo para una página de manual

```
$ man -w find
/usr/share/man/man1/find.1.gz

$ man -w printf
/usr/share/man/man1/printf.1.gz

$ man -w man
/usr/share/man/man1/man.1.gz
```

Buscar una página de manual

Puede buscar `man` páginas que contienen una cadena en particular en su descripción usando:

```
man -k <string>
```

Por ejemplo:

```
man -k unzip
```

Podría volver:

```
man -k unzip
IO::Uncompress::Bunzip2(3pm) - Read bzip2 files/buffers
IO::Uncompress::Gunzip(3pm) - Read RFC 1952 files/buffers
IO::Uncompress::Unzip(3pm) - Read zip files/buffers
PerlIO::gzip(3pm)        - Perl extension to provide a PerlIO layer to gzip/gunzip
gzip(1), gunzip(1), zcat(1) - compress or expand files
IO::Uncompress::Bunzip2(3pm) - Read bzip2 files/buffers
IO::Uncompress::Gunzip(3pm) - Read RFC 1952 files/buffers
IO::Uncompress::Unzip(3pm) - Read zip files/buffers
PerlIO::gzip(3pm)        - Perl extension to provide a PerlIO layer to gzip/gunzip
bzip2(1), bunzip2(1)     - a block-sorting file compressor, v1.0.6 bzcat -
decompresses files to stdout bzip2recover - recovers data from damaged bzip2 files
funzip(1)                - filter for extracting from a ZIP archive in a pipe
unzip(1)                 - list, test and extract compressed files in a ZIP archive
unzipsfx(1)              - self-extracting stub for prepending to ZIP archives
```

Encuentra una página de manual en una sección diferente.

A veces, un término se define en varias secciones del manual. Por defecto, `man` solo mostrará la primera página que encuentre, lo que puede ser molesto para los programadores porque las funciones de C se documentan en una sección posterior a los comandos y las llamadas del sistema. Use lo siguiente para mostrar todas las páginas que coinciden con un nombre:

```
$ man -wa printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man1p/printf.1p.gz
/usr/share/man/man3/printf.3.gz
/usr/share/man/man3p/printf.3p.gz
```

Para ver la página desde una sección específica, simplemente colóquela antes del término:

```
man 3 printf
```

Leer un manual con el hombre.

Esto es lo mismo que leer un manual para un comando:

```
man /path/to/man/file
```

Lea Ver las páginas de manual en línea: https://riptutorial.com/es/unix/topic/442/ver-las-paginas-de-manual

Creditos

S. No	Capítulos	Contributors
1	Empezando con Unix	Community
2	Comandos básicos de la consola	Anand C, Boon, Dirk Schumacher, eli-bd, Jean-Baptiste Yunès, Nathaniel Ford, SarcasticSully
3	Comenzando con los comandos de Unix	eli-bd
4	Descripción general de Unix	Anand C
5	Permisos	Anand C, Chris Forrence, Joey Chatterjee, rama chandra sunkara, William Carron
6	Ver las páginas de manual	Benjamin W., drunken_monkey, intboolstring, Jahid, Simon Jester, Srinidhi, Will