Introducción

Shell

sh

ksh Sustituye a sh (Todo lo de sh y muchas caracteristicas de csh)

csh C shell

jsh Shell de gestión de tareas.

rsh Shell restingido para usuarios inexpertos.

$ Prompt de usuario

% Prompt en csh

prompt superusuario

PS1 Prompt por defecto.

Arbol de directorios

/

6 primarios y 2 secundarios

home

A partir de aquí se crean los directorios de conexión de usuarios (A partir de aquí se crean los arboles de directorios)

Conexión en red

Directamente en UNIX (TCP/IP)

PCTCP (Conexión MS−DOS)

Interface Programa cliente /Independizan hardware y software.

Sistema de archivos

- Se considera cualquier componente un archivo.
- Tipos:
- Ordinarios
- Dispositivos
- Directorios
- Pipes.
- Etc.

Gestión de memoria (Para más información referirise a apuntes de cursos anteriores).

- Paginación por demanda

Seguridad

4 niveles

Nombres de ficheros 14 caracteres (Puede tener o no extensión / Distingue mayusc /minusculas)

INICIO DE SESION

- Contraseña superusuario (root) Privilegios /login/
- Sysadm Usuario administrador que realiza tareas de administración.
- Usuarios Se pone en contacto con superusuario y este le da de alta dentro de la red con un login unico.

Login:

Password:

- Sistema realiza comprobaciones
- Ejecuta el fichero de configuraciones **(.profile)** /Existe uno standard por defecto en el sistema/
- Dependiendo del shell utilizado los ficheros de configuración cambian

Ksh .Kshrc

Mensajes antes de prompt

Motd Mensajes de día

- Permisos por defecto para el root (rw_r__r__)

News Aparecen las ultimas noticias desde la última lectura del indice

You have mail

- Correo nuevo desde la últma lectura.

Posteriormente a parece el prompt ($) y se puede comenzar a trabajar.

Linea de comandos

$Orden −opciones fichero(s)

− opciones Separadas con un flag delante o un unico flag y todas las opciones { Se distinguen mayusculas y minusculas}

fichero(s) Separados por blancos {Admiten anbiguedad}

Ordenes que dan información sobre usuarios.

$who Nombre de usuario , fecha y hora de conexión y otra operaciones.

(para saber si esta abierta la comunicación)

$finger Más información sobre un usuario determinado.

$write

$talk

Conectar dos terminales conectados o abiertos.

Salir CTRL + D

$stty −a Aparecen los caracteres de control configurados.

$mesg −ny Cerrar /Abrir la terminal.

$exit A nivel de usuario se debe salir de forma lógica y cerrar la sesión.

#shutdown −h −y (Automatico) Cierra la sesión de superusuario.

Superusuario.

#wall < fichero

#wall mensaje

- Se envia con esta orden a todos los usuarios conectados.

$ls Lista las entradas de un directorio.

−a Lista ficheros ocultos.

{.nomb−fich}

−ai Nº de i−nodo y nombre del fichero.

−l formato largo.

VARIABLES DEL SISTEMA.

Directorio de conexión HOME

/home/alumnos/fm21/fm21/f951115

$HOME

Directorio de correo MAIL

MAIL = /usr/mail/f951115

Path PATH

PATH = $PATH: $HOME/practica: ...

Prompt PS1

PS1 = Amalia $

Shell por defecto SHELL

SHELL = /usr/bin/ksh

Formato largo {ls} Continuación.

- Tipo de fichero {− d c b p}
- Mascara de atributos rwx rwx rwx
- Nº de enlaces
- Propietario
- Tamaño del fichero
- Fecha y Hora en el formato establecido.
- Nombre del fichero.

−R Recursivo (Listado de todo el arbol que puelga del nivel inicial)

−r Listar en orden inverso

−d Listar la entrada de un directorio (No su contenido)

−F De que tipo es el fichero y lista en columnas

| pg Parada de pagina.

$cat Visualizar el contenido de un fichero.

$cat f1 |pg

$cat f1 > f2

> /dev/lp

$cat f1 f2 f3 Concatena en pantallla.

$cat f1 f2 f3 > f4 El fichero se vacia previamente de contenido antes de concatenación.

$cat f1 f2 f3 >> F1 Se concatena añadiendo al contenido de f1.

$cat f1 _ f2 _ f3 < f4> f5

- Los guiones se toman como entradas atipicas.

Cat Editar ficheros por consola.

$cat > f1

^D

cat _f1 _ > f2

^D

f1

^D

- Se suele utilizar sin opciones.

−v Imprime los caracteres de control

$cat f1 f2 f3 > f4 2>/dev/null {Los mensajes de error se direccionan al dispositivo nulo}

Ordenes de manejo de ficheros ln / cp /mv

- Cuando se realiza un link o se mueve un fichero la información es la misma unicamente se crean nuevos valores de i−nodo o moviendo otro link

$ln D1/f1 D2/f2

$ln D1/f1 /D1/f3 No se crea una copia fisica sino un enlace logico.

- En la orden ls existe un campo que lista el numero de links de un fichero

$mv D2/f2 D2/prac

- Con mv se rompe el enlace antiguo y se crea en la tabla nueva.

$ln usr/bin/ls /usr/bin/dir Realiza un enlace del ejecutable ls al fichero dir

$cp /clase/* **.** {Directorio actual , el destino nunca se puede dejar en blanco}

- La orden mv es equivalente al rem del DOS

$mv D1/f1 D1/f2

$cp D1/f1 /D2/nominas Crea un nuevo fichero con un nuevo i−nodo en destino cambiando sus atributos.

$cp f* nominas cosas otrascosas D2 {Destino}

- Se pueden indicar varios ficheros origen.

$mv f1 fnom D2

$cp −r Copia de forma recursiva.

- Es necesario que esten activos los permisos de lectura en origen y de escritura en destino.

−f Ignora permisos de los ficheros de los que el usuario es propietario

−y Pide confirmación.

$ln −s Realiza un enlace simbolico. {Linka directorios / Sistemas de archivos diferentes}

$ln −s D1/* D2

- Para compartir ficheros es necesario tener activado en el directorio donde se va a linkar el permiso de grupo o usuario para escritura.

$ln D1/f1 /home/alumnos/fn21/pepe /home/alumnos/fm21/luis

- Para borrar un fichero los links no se borran y se ha de ceder la propiedad del enlace para que se pueda borrar.

$chown luis D1/f1

Borrar ficheros $rm

$rm f1

$rm −f Ignorar permisos.

$rm −r Borra de forma recursiva.

$rm −i Pide confirmación.

Ordenes para tratamiento de directorios.

Mkdir Crear directorio.

$mkdir $HOME/D1 D2 /D2/D22

−p Crea una rama entera de un arbol de directorios.

$mkdir −p D2/D22/D222

−m (permisos absolutos en octal) Cambia permisos en el momento de creación.

rmdir Borrar directorios

cd Cambiar el directorio actual.

pwd Directorio actual

Editar ficheros.

$vi (Mirar fotocopias)

Atributos de ficheros.

chmod Cambia los atributos de los ficheros.

umask Establece una mascara de atributos por defecto.

$chmod {u|g|o|a}{+|−|=} fichero(s)

- `=' Asigna en modo absoluto(lo asigna a uno y se lo quita a los demas). {Depende de versones su funcionamiento}

$chmod u+x fnominas

rw− r− − r − −

rwx r− − r − −

$chmod ug+x, g+w fnominas

$chmod g−w fnominas

$chmod u=w fnominas

$chmod 0754 f1

1 Activado bit de archivo para usuario

2 Activado bit de archivo para grupo

4 Fichero bloqueado en memoria (Aparece en ls con una l) ***sticky bit***

$f1 Ej con permiso.

$ksh f1 Invocar al shell para que ejecute el fichero.

−x Ejecución linea a linea.

Umask { Craci ón de una mascara por defecto}

Mascara por defecto

Directorios 777

Archivos 666

$umask

0022 Los que se va a quitar a la mascara por defecto

777 − 022 = 755

666 − 022 = 644

Para que este vigente desde el inicio de la sesión se debe incluir en el .profile.

$chown Propiedad de un fichero.

%chown carlos fnominas

FIND

$find Busqueda de ficheros recursivamente.

$find [origen] −opciones −orden

Ordenes:

−print Imprimir resultados en pantalla.

−exec Ejecutar ordenes.

−ok ejecutar ordenes con confirmación.

Opciones:

−name Busca un patron.

−type

f ordinario

d directorio

c esp. carácter

b esp. Bloque

p pipe

−size { |+|−} nn En bloques de 512 bytes.

−atime +/− nn Accedidos en nn dias.

−mtime +/− nn modificados en nn dias.

−user

−group

−perm (noct) Busqueda de la mascara especificada. (Solo mascara absoluta).

$find $HOME −name *profile* −type f −print − exec {} \;

- Se utilizan las llaves para especificar el nombre de la ruta de acceso actual
- Si se utilizan comodines en el patron de busque debe llevar dobles comillas,

$find −name f* −type f −o −type d −size +6 −print −exec chmod u+x {} \;

Formas de visualizar un archivo en pantalla.

$pg

$more

$tail

$head

Salida paginada.

$cat f1 | pg

$cat f1 | more

$tail f1 Visualiza por defecto las 10 ultimas lineas del fichero.

−n Nº de lineas.

+n Visualiza las n primeras lineas.

−f Visuliza la salidaa que se este ejecutan de findo en este momento.

- Se utiliza para quitar las cabeceras de los ficheros.

$ps > fps

$tail +2 f1

$ps −ef |tail +2 Quita las dos lineas de cabecera.

$head f1 Visualiza por defecto 10 lineas de la cabecera.

$pr Fromatear un fichero por defecto para salida por impresora.

$pr −l12 −w40 −d Cabecera

−l Numero de lineas

−w Numero de col.

−d Doble espaciado.

−h cabecera Sustituye la por defecto y coloca la que indiquemos.

−t Suprime cabecera standard.

Imprimir

Lp Imprimir ficheros.

$lp f1

- Configuración de la impresora Creación de una impresora.
- Direccionamiento Por nombre de impresora o por calse.

−n nº de copias.

−m mensajes con resultado vayan por correo

−t Cambiar la hoja de portada.

−d Indica impresora por defecto.

−w Mensajes a pantalla.

−c Se crea una copia en un buffer para que la impresora imprima el fichero en ese momento por si se dan lugar a modificaciones.

$lpstat −t Estado de la impresora.

−d De una impresora determinada.

$cancel (nº de impresión o nombre de la impresora) Se cancela trabajo de impresión.

- Si se esta imprimiendo en este momento se debe cancelar la impresora.

$lpadmin Admon. De la impresora.

$fmt Formatea el texto al nº de columnas que se le indique.

$fmt −w20 f1

$banner Formato poster el mensaje que se escriba.

Si no se ponen aparece una palabra en cada linea.

$basename Sacar del pathname completo, desde el ultimo separador al final.

$basename $HOME

/home/alumnos/fm21/f953272

$f953272

$dirname Sacar del pathname hasta el último separador.

$dirname $HOME

/home/alumnos/fm21/

Ej .−

$echo Yo soy el usuario `basename $HOME` y estoy conectado en el directorio `dirname $HOME`

``' Operador grae (comillas graves) Ejecuta la orden y sustituye las comills por la salida de la orden.

$cd $HOME/practicas

$practicas= `pwd`

$practicas = $HOME/practicas

$export practicas Exprotar variable a todas los niveles de subshell

Variables del sistema

LOGNAME Nombre del usuario.

$logname Orden equivalente

LPDEST Impresora por defecto.

$cal Saca el calendario del año en curso.

$cal mes año (Hasta el año 2000)

- Se puede indicar en abreviatura o en formato numerico.

$calendar Lee un fichero con nombre calendar a nivel de $HOME, donde reconoce todos los formatos de fecha establecidos y lista aquellos que coincidan con fecha actual.

$echo

Sacar información por pantalla.

Echo $nomvar Devuelve contenido de variables.

Introduce caracteres ANSI

$echo ESC[carácter ANSI]

$echo `Hola \007 Adiod \007 '

$echo \033 [2J Borra la pantalla.

Permite introducir codigos en octal.

Metacaracter Quitar significados del metacaracter del siguiente carácter.

Echo admite caracteres de control

- Caracteres de control de c

$echo Hoy es viernes \n y quedan 3h de clase

$tput Introduce caracteres ANSI

$tput clear Borrar pantalla.

$tput rev video inverso

$tput blink parpadeo

$tput bell bell

$tput sgr0 Volver a video normal.

$setcolor Establecer color.

- echo no necesta dobles comillas
- Si se las indica metacaracter
- Si hay más de una linea se le indica con las comillas.

$date

#date Cambiar formato de fecha.

$date Da la fecha y se permite usar un formato particular.

%m %y %H %s

%d %D %M %T

- Se puede combinar formato con texto y comodines.
- Si es # se cambia para todas los usuarios.

- Si es $ se puede pedir una representación particular.

$date +Hay es dia : %d del mes de %m del año 19%y \n son las %T

Ordenes de busqueda.

$grep Busca un unico patron con o sin ambigüedad en fichero(s).

$fgrep Busca uno o más patrones fijos en fichero(s)

$egrep Busca con ambigüedad uno o más patrones en fichero(s).

$grep `patrones' fichero [{ficheros}..{}]

- Si el patron lleva espacios o comodines debe llevar dobles comillas.

$grep curso* fichUPS /var/UPS/fichcurso

−v Inverso

−i No distingue mayusc /minusc

−l Saca nombres de ficheros.

−n Numeros de lineas.

−c Contador de lineas.

$fgrep `cosas \n otrascosas \n mascosas' ...

- cada patron debe ir en una linea.

$fgrep `cosas

otrascosas

mascosas' f*

- Crear ficherode patrones

$vi patrones

cosas

otrascosas

mascosas

:wq

$fgrep −f patrones f*

^ Principio de linea.

$ Final de linea.

* Cadena de caracteres.

. Un carácter

+ Una o más incidencias del carácter al que precede

| OR logico

! Negación.

$egrep `[Aa].*$\.' F*

- Que empiecen por A o a cualquier cadena cuyo ultimo carácter sea un punto.

$wc

Contador. Cuenta lineas, palabras, caracteres.

−l Solo lineas.

−w Solo palabras.

−c Solo caracteres.

$cat /etc/passwd |wc −l Nº de lineas {Nº de usuarios}

$passwd

new passwd:[ ]

>5 caracteres {Carácter especial letras o numeros}

$passwd

old:

new:

#passwd f9523c Se leda una password.

/etc/passwd {Estructura del fichero}

nombre_usuario:!:UID:GID:información:dir de conexión:shell

! Indica que existe la pasword.

$passwd −s

/etc/shadow

/etc/group Alta en un grupo.

Nombre: clave del grupo: gid: _____ : información.

- Ordenes Pertenecen al tema de administración.

$du Espacio que ocupan en disco o un fichero o un directorio, bloques de 512 bytes, si no se ha indicado ningun directorio o fichero se tomará por defecto el actual.

Todos los ficheros de un directorio estan incluidos, incluso aquellos que esten dentro de los subdirectorios de este.

−a Acepta nombres de ficheros y de directorios como argumentos.

−r Mensajes sobre ficheros que no puedan leerse o abrirse.

−s Proporciona el total acumulado en un directorio concreto y (junto con −a) de todos los ficheros.

$df Da la cantidad de espacio libre de un sistema de ficheros (en bloques de 512 bytes) y el numero de i−nodos libres utilizables en el sistema. El sistema de ficheros puede estar identificado por el directorio de

conexión o por el dispositivo. Si no se especifica se dara información de todos los sistemas de ficheros montados.

−t Ademas del libre informa del espacio restante y el total.

- Un sistema UNIX para que funcione perfectamente debe tener un 10% de espacio libre en el disco.

SHELL'S

- concatenado de ordenes con **;**
- Direccionamiento:

> Direccionamiento de salida (vacia de contenido el fichero destino).

< Direccionamiento de entrada.

>> Doble direccionamiento de salida

<< Doble direccionamiento de entrada

$mail pepe << ! Permite realizar una entrada hasta una clave determinada.

| Pipe Permite introducir como entrada de una orden la salida de otra orden.

- Descriptores:

0 Entrada standar.

1 Salida standar.

2 Salida standar de errores.

$cat f1 f2 f3 > f4 2>ferror

{Concatena f1+f2+f3 f4 | La salida estándar de error ferror.

$cat f1 f2 f3 > f4 2&1 [2 va a 1]

- Variables del SHELL

HOME Dir de conexión del usuario.

PATH Establecer caminos de acceso (separador de caminos `:')

CDPATH Caminos de directorios.

PS1 Prompt primario.

PS2 Prompt secundario.

LOGNAME Nombre usuario.

MAIL Directorio de acceso al correo.

MAILFILE Especificación del fichero de correo.

SHELL Shell por defecto.

TERM Definición del terminal.

TZ Zono horaria.

MAILCHECK Chequea el buzon de correo cada x tiempo

GROUP Nombres de usuarios para grupo activo.

MAILRC Fichero de configuración del correo.

MBOX Fichero donde va a parar correo que se lee o no direcciona.

LPDEST Impresora por defecto.

- Solo tienen valor en el shell de definición.
- Para que tengan valor en otros shell's se tienen que exportar:

$export Exporta a todos los niveles de shell la variable especificada.

Tareas de fondo.

$cat f1 f2 f3 > f4 & {Se indica que la tarea se va a realizar en background}

- Devuelve el numero de identificador de tarea.

- Cuando se sale de la sesión se realiza la orden KILL a todos los procesos.
- Existe la posibilidad de que se siga ejecutando una tarea en background una vez se ha desconectado la sesión. $nohup

$nohup cat f1 f2 f3 |sort > f4

$kill Envia una señal especifica al proceso especificado. La señal se indica con un entero y el proceso con un PID. (Señales son dependientes de la implementación y normalmente se indican el fichero /usr/include/signal.h) Podemos encontrar un PID con el comando **ps.**

$kill 0 Indica la señal de terminación a todo el grupo de procesos en ejecución (excepto el shell que es inmune a la señal terminar) /Normalmente matara a los procesos ejecutados en background/

9 SIGKILL Terminar (No se puede ignorar)

15 SIGTERM Terminar de software (Señal de terminar software)

$nohup Hace que el comando lanzado ignore la señal de colgar que se emite al desconectar la sesión. La salida del comando se redirecciona al fichero nohup.out (colocado generalmente en el directorio raiz).

$nohup comando [comando−argumentos]

Ordenes de control de trabajo.

^Z

bg

fg

jobs Invocar desde cualqier shell (job shell)

kill

stop

- Las ordenes de control de trabajos del shell le permiten terminar un trbajo en modo subordinado(matarlo), suspenderlo sin terminar(pararlo) reanudar un trabajo suspendido en modo subordinado, mover un trabajo de modo subordinado a principal.

^Z Permite suspender el trabajo en modo principal. Esto finaliza el programa y devuelve el control al shell

$jobs Visualizar la lista de los trabajos actuales. La salida muestra los trbajos actuales en modo principal y subordinado, los suspendidos o parados.

$kill Se puede terminar con cualquiera de estos trabajos.

- Se utiliza el carácter % como indicador de trabajo.

$fg Permite reanudar un trabajo en modo suspendido y devolverlo al modo principal.

$fg %2

$bg Reanudar un trabajo suspendido en modo subordinado.

$stop Para la ejecución de un trabajo en modo subordinado, pero no lo finaliza.

Eliminación de significados espaciales en las lineas de ordenes.

- \ Desactiva el significado del carácter especial siguiente y la unión de linea.
- ` ....' Impide que el shell interprete la cadena marcada.
- ... Impide que el shell interprete la cadena marcada excepto $, dobles y simples comillas y \

SHELL KORN.

El shell korn o ksh es otra alternativa al sh. Desarrollado en 1982 por Davis Korn de los Laboratorios Bell.

Proporciona un superconjunto de las caracteristicas del shell del sistema V. Incorpora la mayor parte de las caracteristicas que se incorporan en el Shell C..

Ventaja sobre el csh es que en ksh corren los shells de sh particamente sin modificación.

Presentación e iniciación.

Utiliza dos archivos de iniciación . un solo para presentación y otro cada vez que se ejecute el ksh.

.profile Buscar los ordenes que se quieren ejecutar y las variables y los valores que se quieren que esten activos a lo largo de la sesión.

.kshrc Archivo de entorno especificado en la vasriable ENV (supone que no es un fichero determinado ni de posición fija).

Variables del Shell Korn.

El shell implementa todas las caracteristicas de las variables del shell estándar.

Se pueden definir o redefinir variables, exportarlas al entorno y obtener sus valores. El shell Korn utiliza algunas de las variables del shell del sistema V:

- CDPATH, HOME, LOGNAME, MAIL, MAILCHECK, MAILPATH, PATH, PS1, PS2, SHELL Y TERM

Algunas variables variables importantes utilizadas por Ksh: (.kshrc)

- ENV Localización del archivo de entorno.
- HISTSIZE Cuantas ordenes se guardan en historioco de ordenes.
- HISTFILE Fichero historico de ordenes (.sh_history)
- TMOUT Espera en seg antes de producirse un fuera de tiempo si no se pulsa ni una sola orden.
- VISUAL editor por defecto.
- EDITOR Idem.
- Columns columnas.
- FCEDIT Editor de ordenes Fc.
- FPATH Fichero de autocarga de ordenes.
- LINES Lineas.
- PS3 Prompt para la orden select.
- PS4 Prompt de depuración.

Variables internas del Ksh:

- ERRNO Nº de error de la última llamada fallida.
- LINENO Nº de linea actual.
- OLDPWD Directorio de trbajo antes de realizar la última orden cd.
- PPID PID del proceso padre.
- PWD Dir actual.
- REPLY
- SECONDS Tiempo trnascurrido desde inicio de la sesión.

- Se pueden establecer variables genericas de entorno, y podráin ser cogidas por una orden para ejecutar sin parametros.

- Las variables pueden ser con o sin nombre (REPLY Contendrá los parametros de la última orden definida sin nombre).

$time cat f1 Estaditica de tiempo de ejecución de orden.

- # Nº de parametros definidos como parametros posicionales.

- **?** Valor de retorno de la ultima orden .

- 0 Ejecución correcta.
- Cod error No correcta.

- **$** PID del proceso actual.
- **!** PID del último comando mandado a ejcutar en background.

- **<u>Forma de asignar la salida de una orden a una variable.</u>**

$terminal = $(tty)

$terminal = `tty`

<u>Sustitución de parametros.</u>

${parametro}

$animal = perro

$echo ver los ${aninal}s

${parametro:− valor} Si parametro tiene algun valor sigue con ese valor, pero si no esta definido se define como *valor*.

$bebida= agua

$echo ${comida:−carne} y ${bebida:−café}

$dir= ${1 : − .} Si $1 no tiene valor se toma como valor el directorio actual.

- Se puede introducir un mensaje de error en caso de que una variable no este definida.

$PATH:? NO_PATH}

El `?' indica mensaje de error.

- `=' Si no esta definada o tiene valor nulo se le asigna esa valor
- `+' Si parametro esta definido y no es nulo se cambia por valor, pero si valor es nulo, seguirá siendo nulo.

${#animal} `#' Devuelve la longitud de la cadena.

- De un nombre de fichero con extensión se puede estraer el nombre o la extensión:

$var = abc.123

$echo ${var %.*}

abc

$echo ${var #*.%}

123

Definición de arrays.

- Rango de indices 0 .. 1023

$file[1] = hola

$file[2] = que

$file[3] = tal

Visualizar el contenido de elemento de array

$echo ${file[2]}

Visualizar la longitud del array.

$echo ${#file[*]}

Visualizar todos los elementos del array.

$echo ${file[@]}

Historico de ordenes.

El Ksh mantienen una historia de ordenes. Se puede utilizar esta lista para repasar las ordenes que se introdujeron en la sesión. La lista de historico de ordenes se mantiene a lo largo de las sesiones, de manera que se puede utilizar para repasar o reejecutar ordenes de sesiones anteriores.

Vissualización del historicos de ordenes.

$history

- El último elemento de la lista es la orden que se ejecutó en último lugar.
- El numero de lineas de ordenes del Ksh viene controlado por la variable del shell HISTSIZE.

- Se puede visualizar una orden particular de la historia de ordenes como argumento de history.

$hsitory vi

vi old_note Visualiza la última orden vi.

- El archivo en que se almacena la lista de ordenes viene determinado por la variables del shell HISTFILE.

Reejecución de ordenes.

Se puede ejecutar la orden inmediatamente anterior con la orden ***r (redo).***

$alias r='fc −e −`

- Se pueden ejecutar otras ordenes de la lista de ordenes pasandosela como parametro a la orden r.

$r vi

- Se puede volver varias ordenes atrás en la linea de ordenes.

$r −3 Ejecuta la orden ejecutada hace tres ordenes.(vuelve hacia atrás 3 ordenes en el historico).

Edición de la linea de orden.

Además de ver las ordens anteriores y ejecutarlas, el shell Korn, permite editar la linea de orden actual.

Definición del editor Variables del shell VISUAL | EDITOR

Estableciendo editor por defecto de linea de ordenes $set −o vi

Variable $FCEDIT=/usr/bin/vi

Utilización de caracteristicas de un editor del tipo vi.

El editor opera sobre la linea de orden actual y la lista de historia de ordenes. Cuando se introduce una orden comienza en modo de entrada vi. En cualquier instante se puede entrar en modo de orden pulsando ESC. En modo de orden se puede utilizar las ordenes de edición del editor vi.

Una vez se edita la linea se pulsa INTRO

- Movimiento izq L
- Movimeinto dcha H
- x Borra carácter.
- a Añadir texto.
- i Insertar texto.
- Salta palabras dcha W
- Salta palabras izq B
- Comienzo de linea ^
- Borrar palabra dw
- Reemplazar palabra cw

Manejo del historico de ordenes **$fc**

$fc [−e editor] [−lnr] [primero [ultimo]]

[−e editor] Definir el editor de la linea de ordenes.

[primero [ultimo]] Definir un rango de orden.

−r Invertir el orden de los comandos.

−n No aparecen numeros de orden.

−l Visualiza la lista de comandos.

$fc −l Visualiza los n ultimos comandos (tamaño del buffer)

$fc 15 17 {Rango}

ALIAS

Un alias de orden es una palabra y una cadena de texto que es sustituida por el shell siempre que se utilice esa palabra.

Permite definir un nombre para una orden compleja.

$alias nombre='Orden compuesta'

$alias yo='who |grep amalia'

- Los alias se definen en el shell Korn de la misma forma que se definen las variables.
- Aunque a diferencia de las variables estas hay que definirla y exprotarla a otros subshell, mientras que los alias son reconocidos automaticamente en todos los shell's.

$history $alias history='fc −l'

$alias Lista de todos los alias definidos.

- Para quitar el valor de un alias :

$alias nombre

$alias nombre="

DEFINICIÓN DE FUNCIONES.

Function nombre {orden ; orden; ... ; }

Nombre () {orden ; orden ; ... ; }

- Si se definen en el shell priincipal se quedan cargadas en memoria.
- Se pueden definir en cualwuier shell y exportarlas con **$typeset −f**
- Las ordenes pueden tener ambigüedad o llevar parametros posicionales.

OPERACIONES ARITMETICAS.

$expr expresion Calcula una expresión y nuestra el resultado por pantalla. La expresion puede ser una cadena, un entero o una construcción más elavorada.

Contador = `expr $contador + 1 `

- Los operadores deben ir seguidos y precedidos de blancos.
- Operadores − , * , \ / { Se utiliza el backslash para quitar el valor de metacaracter}

$let Ksh.

- Utiliza los rangos de operadores del C.
- No tener en cuanta los blancos.

$let y = 2*X+5

$y = `expr 2 \* X + 5`

Ordenes para salida y entrada por pantalla y teclado (KSH).

Para imprimir ademas del echo se puede utilizar:

$print <mensaje>

Para leer por teclado (dentro de un guión).

$read −opciones **var1?** Var2 var3 ...

- Var1? Mensaje.

Evaluación de expresiones.

$x = `abc def'

$y = $x

$echo $y

$x

$eval echo $y

`abc def'

$exec

$ksh Versión iniciar sh.

$exec ksh Comineza de nuevo la sesión ksh como shell principal.

~ Sustituye el directorio de conexión.

~/programas

$HOME/programas

/home/alumnos/fm21/f951115/programas

- Puede sustituir solo parte del directorio de conexión.

~/fm21/f951115/programas

Fijación de las opciones del shell Korn.

$set Cambia la configuración o estable nuevos parametros de configuración dentro del ksh.

$set −n Comprueba errores de ejecución.

$set −a Variables son exprotadas automaticamente.

$set −U Con un fichero de ordenes lista las ordenes segu´n se ejecutan.

$set −m Notificación de terminación de procesos en background.

$set −o vi Editor por defecto.

$set −o ignoeeof Impede la despedida accidental de la sesión cuando se pulsa CTRL+D

Con esta opción se debe teclear **exit** para terminar la sesión.

$set −o noclober Impide direccionamientos a un fichero existente.

Si se quiere indicar que realmente se quiere sobreescribir se debe colocar la barra de cauce `|' despues de la redirección.

PROGRAMACION SHELL

$chmod u+x pract1

$pract1

Permisos de ejecución.

$ksh pract1

$ksh −x pract1 Ejecución en modo traza

Ejecución en un shell secundario.

- Para ejecuctar en el shell proncipal se debe indicar con la orden:

$ **.** pract1 { Si no se pone el punto se invoca a un subshell por defecto}

Ej.−

$pwd

HOME/practicas

$(date;cd..;ls)

$pwd

HOME/practicas

- Cualquiere modificación se realiza a nivel subshell y cuando se recupera el shell principal se recupera el entorno anterior, perdiendo las modificaciones de las variables de entorno del subshell

- Para ejecutar ordenes agrupadas a nivel de shell principal se utilizan { }
- Para ejecutar ordenes agrupadas a nivel de subshell se utilizan ( )

Otra forma de ejecutar ordenes:

$ksh

$exec ksh Se coloca como shell principal y el shell anterior desaparece como shell principal.

Conmentarios

No interpretados.

: Interpretados (Sustituye variables por su valor)

Parametros posicionales.

$vi shell1

cp $1/a* .

mv $3/* .

echo Hay $# parametros posicionales

#Para visulizar el numero de parametros pos.

echo $*

#Para visulizar todos los parametros

$chmod u+x shell1

$shell1 D1 D2 PRATICAS

$0 Nombre del programa.

$1 − {99} Parametros posicionales posibles

- Por encima del 9 se deben indicar entre llaves para que se interprete el número completo.

$at Lanzar procesos en diferido.

$at −f fichero [Especificación de tiempo]{Todo lo que no se indique será tomado como actual.

$at −f f1 6am Friday

$at −l Listado de lo lanzado.

$at −r [ ] Retirar la orden {Especificación del ID de la orden obtenida en at −l}

- No todos pueden usar el comando at:

- El superusuario crea un fichero para saber quien puede usar el fichero o no .

/etc/cron.d/at_allow

/etc/cron.d/at.denny

Donde se encuentra un nombre de usuario por linea.

- Si no existen estos ficheros solo lo puede usuar el root.
- Si existe el segundo de los ficheros pero esta vacio todos pueden usar el comando at.
- Solo puede existir uno de los ficheros a la vez.

DEMONIO Proceso que crea un usuario, superusuario o sistema y que una vez lanzado es gestionado por el sistema y ya no esta bajo el control del usuario que lo lanzó.

$vi demonio−1

cd ${PAPELERA:− $HOME/junk}

rm −r

at midnight Friday << **!** {Direccionamiento de entrada hasta que encuentre el carácter indicado}

demonio−1 &

!

:wq

$chmod u+x demonio−1

$demonio−1 &

- Se puede utilizar el cron (demonio del sistema) para ejecutar este tipo de ordenes sistematicas.

$touch fichero En el modo−i se mantiene información sobre el momento de creación, modificación y acceso de un fichero. Por defecto el comando touch actualiza las fecha de modificación y acceso, con información temporal especificada o fecha y hora del sistema.

Por defecto, un fichero nombrado que no existe se crea.

−a Actualiza solamente horas de acceso.

−m Actualiza horas de modificación.

−c No crea fichero si no existe.

- La roden set nos permite establecer variables de entorno

$set var =`ls −n`

- Si se da una orden sin argumentos se toman por defecto los del entorno.

$echo $var

f1 f2 ....

$1 $2 ...

- Las variables pueden no tener nombre: {$reply}

($*) Referencia a la variable sin nombre.

- Solo puede haber una variable sin nombre.

$set fecha= `date + −−−−−−− `

$trap Esta orden permite espeficar una secuencia de ordenes que se ejecutan cuando se reciben señales de interrupción en los programas shell.

Formato general:

$trap {`orden; orden; ...; orden'} señales

- El primer argumento de trap se toma como la orden u ordenes a ejecutar. Si se tienen que ejecutar una secuencia de ordenes se han de incluir entre comillas simples
- Las señales son los codigos de las interrupciones. Las más importantes son:

- 0 Salida del shell.
- 1 Colgar
- 2 Interrupción, DELL
- 3 Salida.
- 9 Matar (No puede ser interrumpida).
- 15 Terminación (Kill normal).

$trap `cat .ch_history>>$home/ordenes.his; rm .sh_history' 0 1 3 15 {Posibles salidas del sistema}

Exit Cuando un proceso termina devuelve un valor de salida a su proceso padre. La orden exist es una orden de shell interna que hace que el shell slga y devuelva un valor de estado de salida. Por convención el estado 0 significa terminación normal, y un estado distinto de 0 significa que sucedió algo anormal

$xargs Una de las caracteristicas más utilizadas de la programación es la de conectar la salida de un programa con la entrada de otro utilizando cauces. A veces se utiliza la salida de una orden para definir los argumentos de otra

Es una herramienta de programción shell que permite combinar la entrada de una tuberia o cauce con la entrada de parametros posicionales de un shell script.

$xargs[indicadores] [orden (argumentos iniciales)]

- Toma los argumentos iniciales , los combina con los arguemtnos leidos desde la entrada estandar y utiliza la combinación para ejecutar la orden especificada.

$ls $1 | xargs −i −p mv $|() $2()

$find : −name a* −type f −exec rm {}\;

$find −type f −print |xargs grep −l −i $@

$* { Todos los argumentos de la linea de ordenes.

- Funde la entrada de la orden find.

$echo $* | xargs −n2 −p diff

{ Argumentos de dos en dos}

SENTENCIAS DE PROGRAMACIÓN

Orden if

If orden

Then

Orden(es)

Fi

Else

Orden(es)

Fi

- La condición puede ser una orden o sentencia test.

$if orden ; then ordenes; ...{; ...;} ; else orden; ...; fi

- Se puden anidar.
- La combinación else if elif

If orden

Then

Ordenes

Elif orden

Then

Ordenes

Ej

If mkdir −p $HOME/D1/D11

Then

Echo Creación correcta.

Else

Echo No se pudo crear directorios

Exit 1

Fi

Orden Test

Permite comparar ficheros, números o cadenas.

If test −w f1

Then

Rm f1

Else

Echo fichero protegido

Fi

Opciones de test:

- Para ficheros:

−a Existe

−r Exist y perm lect

−w Exist y perm escritura.

−x Exist y perm ejecución

−f Exist y fich ordinario.

−d directorio

−h Simbolico.

−p pipe

−s Si existe y tamaño mayor que cero.

−c Disp orient. a carácter.

−b disp orient a bloque

- Para números:

N1 −eq N2

N1 −ne N2

N1 −gt N2

N1 −ge N2

N1 −lt N2

N1 −le N2

- Cadenas

Cadena1 = Cadena1

Cadena != Cadena2

−z Existe y longitud es 0

−n Existe y long es distinta de 0

cadena Si existe.

- Combinaciones:

−o OR LOGICO.

−a AND LOGICO

Ej

If [−w f1] Sustituye a test.

- En Ksh

[[n1 >= n2]] Se pueden usar caracteres de comparación.

Bucle For

For i

in lista

do

orden(es)

done

- La variable i puede ser cualquier nombre que se elija
- Si se omite la parte in lista de la sentencia el bucle se ejecutará una vez para cada parametro posicional.
- Tambian se puede especificar la repetición un número determinado de veces especificando este en la lista.

For i in 1 2 3 4 5

Do

Orden(es)

Done

- Se pueden anidar bucles

For i

Do

For var

In 1 2

Do

Proof $i

Done

Done

- Cuando se le indica un asterisco en la lista si hay variables definidas se utilizan como entradas sichas variables y si no se utilizan las entradas del directorio actual.

Sentencias While y Until

While ordenes1

Do

Ordenes 2

Done

Until orden

Do

Orden(es)

done

while true

do

done &

until false

do

done &

Ej.−

While true

Do

If [−n $MAIL]

Then

Echo You Have mail

Fi

Sleep 30

Done &

Sentencia Shift

$shift Rueda de partametros. Salta parametros a la derecha.

Sentencia Case

Case $var in

- ordenes

;;

- Ordenes

;;

*) Ordenes

;;

esac

- En el patron se puede usar todo tipo de comodines.

Sentencia Select

- La orden slect visuliza una serie de elementos sobre el error estandar y espera entrada.
- Si el usuario pulsa INTRO sin realizar seleccción la lista de elementos se visualiza de nuevohasta que se le da una respuesta.
- Orden especialmente apropiada para la realización de menus

- Definir el prompt para el select

PS3 = ¿Qué opción desea?

Select i | var in att cancept dec hp

do

case $var in

att) TERM=360

break

;;

concept) TERM=C108

break

;;

dec) TERM=vt100

break

;;

hp) TERM=hp2612

break

;;

esac

done

export TERM

Sentencia Exit

Salida del programa al shellprincipal, independientemente del subshell en que se encuentre.

Sentencia Break

Sale de unnúmero determinado de shell's. El numero de shell's le es indicado con un parametro entero.

Tambien hace salir del bucle que lo engloba inmediatamente o el numero de blucles que le indiquemos con el parametro entero.

Break nn

Sentencia Continue

Salta iteraciones dentro de un bucle.

Slat el número de iteraciones que se le indique con el parametro entero y por defecto es una.

Cut

$cut −c list [fichero(s)]

$cut −f list [−d char] [−s] [fichero(s)]

En este caso list identifica las columnas o campos que deben pasar a la salida y char es el carácter que se utiliza como delimitador de campo.

- La opción −c del comando procesa las columnas que le ordenan entre una lista de ficheros y muestra los reaultados por pantalla. Una columna es un carácter en una posición determinada de la linea.
- El carácter tab es el delimitadar de campo por defecto.
- Opción −d permite cambiar el delimitador de campo por defecto.

Paste

$paste [−dlista] ficheros

$paste −s [−dlista] ficheros

El comando tiene dos modos :

- Fusión paralela Toma una lista de ficheros de entrada como argumentos, fusiona las lineas correspondientes de cada fichero y envia el resultado a la salida estandar. Las lineas se fusionan con el carácter TAB por defecto.
- Fusión serie Opción −s. Cada linea de entrada se añade al final de la linea anterior generando así lineas más largas

Join

$join [opciones] fichero1 fichero2

- Toma dos ficheros como argumentos.
- Normalmente el primer coampo de cada fichero se toma como campo de unión.El comando muestra el campo de unión seguido del resto de la linea del primer fichero y el resto de la linea del segundo

fichero.

Sort

$sort [opciones] [fichero(s)]

- Toma como entrada una lista de ficheros de texto ordenandolos por lineas, y envia el resultado a la salida escogida. Si se suministra más de un fichero de entrada los ficheros se fusionarán en el proceso de ordenación.

−c Comprueba que el fichero de entrada este clasificado.

−m Fusiona los ficheros clasificados suponiendo que los ficheros de entrada ya están ordenados.

−u Suprime todas las lineas con identica clave de clasificación excepto una . (Por defecto toda la linea)

−o salida Fichero de salida.

−f No distingue mayusculas y minusculas.

−n Ordena numericamente.

Ej .−

$sort f1 f2

$sort −m f1 f2

$sort +2 −5 f1 Prdena por los campos 3,4,5

$sort +3.2 −5.7

Int {caract 3.3 a 5.7}

- Con la clausula UNIQ Aunque no sea campo clave suprime duplicados.

Uniq

$uniq −udc [+m] [−n] [entrada [salida]]

- Lee la entrada y compara las lineas adyacentes. Si dos o más lineas son iguales se borraran todas menos una.
- −u Salida a aquellas que no esten repetidas.
- −d Solamente a una copia de cada linea repatida.
- −c Indica el número de repeticiones.

Tee

$tee [−i] [a] [fichero(s)]

Transmite la entrada estandar a la salida estandor a la vez que realiza una copia de la salida en un fichero.

−a En lugar de sobreescribir añade.

Ej.−

$ls −Ra | tee −a listados | tee /dev/tty24 | tee /dev/lp0 |wc −l

Ej .− Progrma que analice las entradas de un directorio que se trata como parametro y analice si un fichero es ordinario, si tiene permiso de lectura lo liste, si es un directorio muestre su contenido y si es un especial muestre un mensaje.

Vi shell1

If [s# −ne 1]

Then

Echo Programa necesita un parametro.

Exit 1

Fi

#Opcional

If [s# −me 1]

$1 = `pwd`

fi

#Comprobación del numero de parametros

case $# in in

- echo Debe haber por lo menos un parametro.

Exit 1

;;

- echo Correcto

;;

*) echo numero de parametros incorrecto

;;

for i in `ls $1`

do

```
if [−f $i −a −r $i]

then

cat $i

elif [ −d $i ]

then

ls $i |pg

elif [ −c $i ]

then

echo El fichero $i es especial orient a caracter.

Elif [ −b $i ]

Then

Echo El fichero $i es especial orient a bloque

Else

Echo tipo desconocido

Fi

Done
```

Ej .− Directorio varios que cuelga de la raiz, donde existen ficheros propiedad de diferentes usuarios. Se pide construir un shell que copie dichos ficheros en el directorio de conexión de los propietarios eliminando al final el directorio varios.

```
For i in `ls −l |tr −s   : `

Do

Prop=`echo $i |cut −d: −f 3 `

Fichero= `echo $i | cut −d: −f 7 `

Mv /variso/$fichero ~/$prop

Done

Rmdir /varios
```

Crear un demonio que informe semanalmente de los usuarios que se han dado de alta o de baja en la ultima semana

Procedimiento que se lance todos los dia al finalizar la sesión que analice el directorio de conexión borrando ficheros duplicados, ficheros vacios o directorios vacios.

AWK

Incorpora el lenguaje C completo.

Lee la entrada procedente de un archivo, un pipe o teclado. Busca en cada linea el patrón o patrones especificados y ejecuta las acciones correspondientes.

$nawk `/patron/ {acción} ` fichero

$awk −opciones `/patron/ {acciones} ` fichero

−Fc Hace el carácter c como separador de campos.

−v var = ... } Define una variable.

−f patron Fichero de patrones

- Se puede omitir el patrón o las aciones

- Si se omite el patrón se realiza la acción sobre todas las lineas.
- Omite la acción Lista lineas con el patron indicado.

Awk separa cada linea automaticamente en campos. Siendo:

$1 Primer campo

$2 Segundo campo.

${199}

Siendo S0 La linea completa.

- Los campos van separados por el separador de campos que por defecto es el espacio en blanco.

Especificación de patrones:

- Patrones anteriores (egrep)
- ~ Incluido

$2 ~ /^15/

$2 == $3 Comparación.

$1 != $3 Neg condicional.

$1 = cadena o Número Asignación.

$2 > Juan

- Patrones compuestos:

&& y ||

- Patrones de rango separados por ;

/^200/; /^299/

BEGIN

Establecer acciones a priori

Vi f1

BEGIN {

If (ARGV[1]==)

Datos = datos.dat;

Else

Datos = ARGV[1];

ARGV[1] = ;

}

}

$awk −F: `BEGIN {} /patron {acciones}' f1

$awk −F, `**END** {print NR } inventario

NR Número de registros leidos.

La sentecia print Puede usarse el formato de C o de UNIX.

Si se quiere implimir varios campos

Print $1 $2 $3

Separador Blanco , ; o cadena

Comentarios #

Varibles

- De usuario No necesitan def de tipo previa.

- Son variables numericas o de cadena.
- Inicialmente puestas a nulo.

- Globales al programa awk excepto dentro de las funciones definidas por el usuario.
- Internas Se fijan automaticamente y tiene valor por defecto estandar.

FS Separador de campo de entrada.

OFS Sperador de campo de salida.

NF número de campos del registro actual.

NR Númerod e campos leidos hasta el momento.

FILENAME Nombre de archivo de entrada.

FNR Nº de reg del archivo actual.

RS Separador de regsitro de entrada.

ORS Separador del registro de salida.

ARGC Nº de argumentos.

ARGV Ptros a argumentos.

ENVIRON Array de variables de entorno

Definición de arrays.

Igual que en UNIX Asignación.

Borrar elemento

{DELETE nomarray[nro id]

Se pueden definir anidados.

Definición de sunciones de usuario.

Function nombre (lista de parametros)

{

acciones

return (valor);

}

- Pueden ser recursivas.

Ej.−

```
$vi prog

BEGIN{

If (ARGC != 2)

Printf (\n error \n);

Else

{

num = ARGV[1];

ARGV[1] = :

};

printf (El factorial de %d es : %d  , num, factorial (num));

}

function factorial (n)

{

if (n <= 1)

return 1

else

return (n* factorial(n−1))

}
```

Llamada $awk −F −f prog 3

Funciones internas.

Aritmeticas.

Exp(n) Exponencial.

Log(n) log en base 10

sqrt(n) Raiz cuadrada

sin(n)

cos(n)

atan(n)

Funciones de cadena

Longitud del argumento asociado (por defecto $0) **lenght (arg)**

Rand Devuelve un numero aleatorio

Srand Permite cambiar la semilla (Prodefecto la hora)

Int(n) Trunca a entero.

Index(cadena, subcadena) Devuelve pos donde se encuentra subcadena.

Match (cadena, exp regular) posición de la cadena donde se encuentra exp.

Splip (cadena, array, sep) divide cadena en un array con tantos eleemntos como determine el separador de campos.

Sub (exp regular, subcadena, cadena) Sustituye en cadena la primera ocurrencia de exp regular por la subcadena.

Gsup (exp regular, subcadena, cadena) Sustituye todas las ocurrencias devuelve nº de cambios.

Toupper(cadena) minisculas mayusculas.

Tolower (cadena) mayusculas minusculas

Substr (cadena, m ,n) Toma de cadena el numero n caracteres a partir de pos m.

System (orden) Ejecuta orden de UNIX

Forma de leer por consola

Getline var < fichero {Lee fichero linea a linea}

Getline var < /dev/ttyp1

Orden | getline var

Ej.−

PRODUCTOS	**ENERO**	**FEBRERO**	**MARZO**	**ABRIL**	**MAYO**	**JUNIO**
PERAS	30	10	20	15	20	0
UVAS	10	10	20	30	50	0
MANZANAS	10	10	10	10	10	10

- Consumo total por producto
- Consumo total por mes.

Vi prog

```
BEGIN {FS=\t}

NR=1 {NUMCOL=NF

Printf ( %s \t TOTAL \n, $0);

}

NR!=1 {

Total_producto = 0;

For (i=2 ; i <= NF; i++)

{ total_mes[i]= $i;

total_producto += $i

}

total += total_producto;

printf ( %s \t %d \n, $0, total_producto);

}

END {

Printf (TOTAL \t);

For (i= 2, i<= NUMCOL, i++)

printf (%d \t, total_mes[i]);

printf (%d \n , total);

}
```

$awk −f prog hoja

· El fichero hoja es el contienen la tabla anterior.

Ej.− Mediente awk simular, dar de baja un usuario.

Este fichero presenta el esquema basico para dar de baja un usuario. Tiene como parametro el nombre del usuario.

- Crear las variables con las caminos de directorios.

```
PASS= /etc/passwd

SHAD=/etc/shadow
```

```
PASST=/tmp/passwdtempdel

LOCK=/temp/ptmp

CORREO=/var/mail

Vi userdel.

# Comprobar que el numero de parametros es correcto

case $# in

1);;

*) echo sintaxis userdel : userdel nomb_usuario 1&2

esac

if test ! −w $PASS

then Permiso denegado ; exit 2

fi

if test −a $LOCK

Then echo Fichero $PASS bloqueado; exit3

Fi

Touch $LOCK

Trap rm −f $LOCK; exit 10 1 2 3 15

If grep −s $1 $PASS

Then echo .

Else

Echo El usuario $1 no existe;

Rm −f $LOCK; exit 5

Fi

Raiz =$(egrep $1 $PASS | cut −d: −f 6);

If test −d $raiz

Then rm −r $raiz
```

Else

El directorio raiz no existe ; rm .f $LOCK ; exit 6;

Fi

Grep −v $1 $PASS > $PASST

Mv $PASST $PASS

Grep −v $1 $SHAD > $SHADT

Mv $SHADT $SHAD

Rm $CORREO/$1

Rm lf $LOCK ; exit 0

PROCESOS

Un proceso puede ser uno o más de un programa en ejecución.

El termino proceso fue utilizado por primera vez por los diseñadores del sistema MULTICS (Antecedente del UNIX).

Siempre que se crea un proceso en UNIX se utiliza la orden FORK para bifurcar un proceso en proceso padre e hijo (Independiente).

Los procesos son entidades con vida propia.

Orden ps (porcess status).

Lista todos los proceos activos en ejecución en la máquina.

$ps Información referente a los proceos asociados con un terminal.

Los ID de los procesos se asignan de forma consecutiva a partir de :

- El proceso 0 es un proceso del sistema que se crea al principio cuando se crea el sistema UNIX.
- El proceso 1 (INIT) es el responsable de la generación de los restantes procesos.

Hasta el rango de los numeros enteros pos y cuando se terminan se asignan los PID de los procesos liberados.

- Generación de los procesos en arranque.

- Cuando se inicia o arranca el sistema UNIXm el nucleo (/unix) se carga eb memoria y se ejecuta.
- El nucleo inicia las interfaces hardware y las estructuras de datos internas y crea un proceos de sistema (proceso 0 swapper). El proceos 0 se bifurca (FORK) y crea el primer proceos de usuario (proceso 1 INIT)

Eliminar un proceso.

Terminar un proceso

$kill {ID del proceso}

- Envia una señal al proceso. Cuando se utiliza sin argumentos la orden kill envia la señal 15 al proceso (Terminación software).
- Existen procesos exentos a esta señal, estos proceos se eliminan utilizando la señal −9 (Eliminación incondicional).
- Eliminan todos los procesos creados creados durante una sesión.

$kill 0

Planificación de Procesos.

$at Especificación del momento de ejecución de una orden.

- Permitidos formatos de fecha/hora.

−f Ejecutar una secuencia de ordenes contenidas en un fichero.

−l Devuelve el ID y el tiempo de planificaciónde todos los procesos lanzados con at.

−r Suprimir un trabajo de la cola.

$batch Permite diferir ejecución de una orden pero sin controlar cuando se ejecuta.

- Especialmente util cuando se dispone de un conjunto de ordenes y no importa el momento de ejecución.
- La salida y el error se envian con mail (Excepto que se indique lo contrario).

- Procesos lanzados tiene una prioridad inferior a background.

Demonios Son proceos que no estan conectados a un terminal, pueden ejecutarse en modo subordinado y realizan un trabajo util para un usuario.

Cron.− Demonio del sistema que ejecuta ordenes en instantes especificos. La orden y la información de planificación estan mantenidos en el directorio

/var/spool/crontabs

/usr/spool/cron/crontabs (Ant .− SVR4)

- Cada usuario que esta titulado para palnificar directamente ordenes de cron tiene un archivo crontab.
- Cron despierta periodicamente y ejecuta trabajos que esten planificados para ese instante.

Estructura del fichero crontab

MIN HOUR DOM MOY DOW COMMAND

Separador Espacio en blanco.

- El archivo /etc/cron/cron.d/xcron.allow contiene la lista de todos los usuarios que tienen permitido

planificar procesos

$crontab Añadir lineas al fichero crontabs.

$crontabs

<linea con formato >

^D

Borrar el crontabs

$crontabs

^D

{Error frecuente}

Prioridad de procesos.

A los proceos en UNIX SV se les asigna recursos para ejecución secuencialmente.

Como se planifique la ejecución depende de la prioridad asignadad al proceos.

Los procesos del sistema tienen una prioridad superior a la de todos los proceosos de usuario.

La prioridad de todos los procesos de usuario depende de la cantidad de tiempo de CPU usado.

Prioridad Proceos interactivos tienen una mayor prioridad frente a los procesos de gran consumo de CPU.

- UNIX no permite mucho control sobre la planificación de procesos aunque se puede influir mediante la orden NICE.

$nice Permite ejecutar una orden con una prioridad más baja de lo normal.

Prioridad = Indice de prioridad + valor de nice { Disminuye valor de prioridad}

{10 unidades por omisión}

$nice −nn (Disminuir) {comando}

$nice − nn (Aumentar)

- A nivel de usuario se puede disminuir la prioridad de un proceso mientras que es solo el superusuario quien puede aumentar este valor.

- Solo se puede hacer en procesos de tiempo compartido.

$sleep La orden no hace nada durante el tiempo indicado en seg.

/uso en guiones/

$sleep {tiempo en seg}

while true

do

date

sleep 30*60

done &

$wait Sincronización de ordenes en programas shell que se ejecutan asincronamente.

- El proceso padre espera a que finalice el proceso hijo.

Opciones de ps

−f Listado completo de procesos

UID

PID

PPID

C Indice de utilización de CPU.

STIME Tiempo desde el inicio de la ejecución.

TTY Terminal asociado.

TIME Tiempo acumulado de CPU.

COMMAND

−l Formato largo

Campo **F** Caracteristicas del proceso

Campo Estado de un proceso

O P. Ejecución.
S P. Durmiendo.
R Ej en cola
I Inactivo en creación.
Z zombie
T P detenido y en modo de traza.
X P a la espera de más memoria.

Campo **PRI** Prioridad de procesos .

Campo **NI** Valor de NICE

ADDR Dirección inicial en memoria.

SZ Tamño en paginas.

−e Visualización de todos los proceos del sistema.

Procesos en tiempo real.

Clases de prioridad de procesos

Tiempo Real

Tiempo compartido (Planificación propia)

Clase del sistema (No planificacbles)

Valores de prioridad.

Prioridad en tiempo real (0−X)

X Mayor valor permisible.

Proceso con la prioridad más alta es el proceso de tiempo real con el maximo **rtpi** (Prioridad siempre inferior a la clase del sistema).

Especificación de prioridad.

#priocntl Modificar parametro de prioridad.

#priocntl −s [−c clase] [opciones de clave] [−i tipo] [id lista]

−s Convertir.

−c clase Indica la clase de prioridad. (RT Tiempo real)

Ejecución de un proceso con una clave de prioridad.

−e Permite lanzar un proceso nuevo con una prioridad determinada

−t [miliseg] controlar el cuanto de tiempo en la clase de tiempo real.

−d Visualizar parametros de planificación de procesos.

−l Determinar que clases de procesos estan soportadas por el sistema.

Señales.

Soporta 32 señales

−9 muerte incondicional.

−15 terminación software.

$nohup seguir ejecutando proceos cuando se termina la sesion (Ignorar la señal de hangup)

$trap Capturar señales.

Getline var < terminal

Getline var Leer por teclado.

Ej.−

```
Echo −−−−− AGENDA −−−−−−−−
PS·= `Elija una opción: `
If [ ! −f agenda ]
Then
Touch agenda
Fi
While true
Do
Tput clear
Select op1 in ACTUALIZAR CLASIFICAR SALIR
Do
Tput clear
Case $op1 in
ACTUALIZAR)
While true
Do
Select op2 in Altas bajas Modificaciones conultas salir
Do
Case $op2 in
Tput clear
```

```
Altas) alt

Break ;;

Bajas) clave

Baj $?

Break ;;

Modificaciones) cave

Mod $?

Break ;;

Salir) break 3;;

Esac

Done

Done

Break;;

CLASIFICAR) clave

Let y=$?

Let x= $?−1

Touch agenda.ord

If [ $y = 1 −o $y= 5 −o $y=6 ]

Then

Sort −n −t: +$x −$y agenda > agenda.ord.

Else

Sort −t: +$x −$y agenda > agenda.ord

Fi

Mv agenda.ord agenda

Cat agenda

Break;;
```

```
SALIR) exit 0

Break ;;

Esac

Done

Done

Vi clave

PS·='Elige clave: `

Slect in NIF NOMBRE APELLIDOS DIRECCION COD_POS TELF

Do

Case $i in

NIF) exit1 ;;

NOMBRE) exit 2;;
APELLIDOS) exit 3::
DIRECCION) exit 4;;

COD_POS) exit 5;;

TELF) exit 6;;

Esac

Done

Vi alt

Terminal= $(tty)

Awk `BEGIN {FS= :; OFS=:; print Introduce entrada con formato: NIF(9); NOMBRE(10) ...

Term=ARGV[1]; ARGV[1]=;

Getline registro < term

Print registro >> agenda }' $terminal

Vi baj

Terminal = $(tty)

Touch agenda.temp
```

```
Awk `BEGIN { FS=:; OFS=:; print Introduce ka clave a dar de baja \n

Term= ARGV[2]; campo= ARGV[1]; ARGV[1]=; ARGV[2]=;

Getline clave < term}

$campo !~clave { print >> agenda.temp}' $1 $terminal agenda

mv agenda.temp agenda
```

SEGURIDAD

El sistema UNIX fue diseñado de modo que los usuarios pudieran acceder facilmente a los recursos y compartir información, por lo que la seguridad era un aspecto secundario.

En versiones recientes del sistema se han ido incluyendo mejoras en la seguridad que hacen más dificil la obtención de acceso por parte de usuarios no autorizados.

Archivos de contraseñas.

Dos archivos /etc/passwd y /etc/shadow . Utilizados por el programa login para validar los usuarios y para preparar el entorno de trbajo inicial.

/etc/passwd

- Accesible por todos los usuarios.
- Existe una linea por cada usuario del sistema y para ciertos nombres de presentación del sistema.

Campos:

- Nombre de presentación.
- Segundo campo X (Indica si existe contraseña)
- UID
- GID
- Comentarios Información general del usuario.
- Directorio propio $HOME
- Shell de presentación.

- El root en el etc/passwd esta contenido en la primera linea del fichero. Como UID tiene el valor 0 y su directorio de conexión es el raiz / y su shell de presentación es sh (shell estandar).
- Existen determinados nombres de presentación del sistema utilizados por determinadas ordenes para la gestión y administración del sistema.

/etc/shadow

Campos:

- Nombre de presentación.
- Contraseña cifrada de ese nombre de presentación (NP No contraseña).
- Nº de dias entre 1−1−70 y el dia de la ultima modificación de contraseña.
- Nº de dias minimos requeridos para modificaciónes en la contraseña.
- Nº max de dias que la contraseña es valida

- Nº de dias antes de la expiración de la contraseña.

- La utilización de /etc/shadow se basa en que todos los usuarios tienen acceso al /etc/passwd mientras que solo el root puede leer el shadow Evitar el acceso directo a la clave de cifrado.

Cifrado de archivos.

Se puede desear mantener algunos de los archivos de un usuario cifrados (incluso para el root). Puede proteger estos ficheros cifrandolos.

Cuando se cifra se utiliza un proc que cambia el contenido del fichero por otro aparentemente sin significado. (texto cifrado)

Crypt cifrar archivos.

Es necesario suministrar una clave de cifrado

a) Bien como argumento de la orden(desaconsejado por ser visible con ps)

b) Respuesta a un inductor.

c) Variable de entorno (CRYPTKEY)

Opcion −k

Para desencriptar un fichero se utiliza el mismo metodo de encriptado sumistrando la clave de cifrado por ser el procedimiento el mismo en ambos sentidos.

Compresion de archivos.

La compresión remplaza el contenido de un fichero por una version codificada con menor numero de bytes.

El archivo original puede ser recuperado utilizando un metodo de descompresión.

SV proporciona dos ordenes para compresión de archivos.

$pack Archivo comprimido tiene el mismo nombre pero con ext **.z**

Utiliza el error estandar para informar del resultado de la compresión.

$unpack Recupera el archivo original a partir de la versión empaquetada.

- La orden opack Codificación Huffman.

$compress Mejora el algoritmo de compresión.

Cuando se ejecuta creara el archivo con extensión **.Z**

$uncompress Recupera la versión original.

- A diferencia de pack no emite información acerca del resultado del proceso de compresión y descompresión.

- Para visualizar la versión comprimida de un fichero utilizar

$zcat archivo.z

Permisos Suid y Sgid

#Explicación de los modos 1000 /2000 / 4000 /6000 de la mascara de atributos del fichero.

Especificación de permisos de suid y sgid

Mediante :

$chmod u{+/−}s fichero

$chmod g{+/−} archivos

- Se puede modificar estos valores suministrando una cadena de 4 digitos en octal a chmod donde el digito más a la izquierda indica los permisos de suid y sgid.

Shell Restringuido(rsh).

Proporciona al usuario capacidades restringuidas.

Los administradores de sistema pueden evitar que el usuario utilice determinados programas asignando este shell restringuido, como programa de inicio.

Se puede invocar desde la linea de ordenes ejecutando

$sh −r

Restricciones:

- Los usuarios no pueden trasladarse de su directorio propio (cd desactivado).
- No se puede modificar el valor de PATH Limita ejecución de ordenes a las contenidas en los directorios indicados en PATH.
- No se puede modificar var SHELL
- No se pueden ejecutar ordenes en directorio diferente al actual.
- No se puede redirigir la entrada (> o >>)
- No se pueden utilizar la orden exec.

- Utiliza el mismo programa que el sh pero al ejecutarlo se le aplican las restricciones.

ADMINISTRACION

Superusuario La mayor parte de la administración se realiza por parte del susperusuario.

Su nombre de presentación es el **root .**

Como superusuario se tiene control completo sobre los recursos de la computadora.

El introductor del root es la almohadilla #

Puede cambiar la fecha y la hora del sistema y la especificación de zona horaria.

Mediante la orden #date y la variación de la variable de entorno TZ.

Es el encragado de asignar un nombre al sistema que será utilizado para la identificación de este de cara a las comunicaciones con otros equipos.

#setuname −s

#setuname −a Información del nodo.

- Otra tares que realiza el sistema es el arranque y la desconexión.
- El sistema tienen diferentes estados de arranque o estados de INIT. Estos permiten que el superusuario pueda limitar ciertas tareas cuando se realizan ciertas tareas administrativas.

El sistema tienen un estado por omisión que se guarda en el archivo /etc/inittab, que contiene una linea con INITDEFAULT y un nº con el INIT por defecto en arranque (=2 Multiusuario en red local).

Durante la sesión se puede cambiar el INIT con la orden:

$init nn

El el Shutdown La opción −i permite establecer el INIT del siguiente arranque sin tocar el INITDEFAULT

Valores INIT.

- 0 Estado de desconexión o máquina desconectada. Diagnosticos de hardaware.
- 1 Estado monousuario o administrativo (Anula procesos terminales) . Cambias en la configuración.
- S o s Equivalente al 1. Diferencia es que los sistemas de archivos no estan montados (No hay ni archivos ni procesos).
- 2 Multiusuario. Modo normal de operación. INIT por defecto.
- 3 Para poder usar el sistema remoto de ficheros. Se puede arrancar directamente o al añadirse los sistemas RFS (Remote File System).
- 4 Estado definido por el usuario.
- 5 Estado firmware. Para hacer pruebas especiales. Arranque del sistema desde diferentes archivos de arranque.
- 6 Estado de parada o arranque.
- a,b,c Pseudo estados. Dentro del inittab se pueden definir pseudoestados, ejecuta el pseudo estado y vuelve al INIT por defecto.
- Q Si se realizan modificaciones. Lee el archivo inittab y se desea iniciarlo sin cambiar el estado actual.

Shutdown Orden de desconexion del sistema y se utilizan tambien para pasar a un estado más bajo. La razon de su utilización es la posibilidad de conceder un periodo de gracia a los demás usuarios antes de desconectar la máquina.

#shutdown −y −g{tiempo}.

Visualización del entorno de usuario.

Antes de añadir un usuario al sistema se deberia mirar los valores por defecto de creación de usuarios. Estos valores muestran la información utilizada para crear un usuario en el sistema.

#useradd −D Muestra los valores por defecto de creación.

Si se quiere modificar el entorno

−g grupo.

−d directorio.

−e tiempo de expiración de la clave.

Cuando un usuario se presenta al sistema se ejecutan dos archivos como parte del perfil de un usuario, el /etc/profile que se ejecuta para todos los usuarios del sistema y el .profile propio de cada usuario.

El .profile del directorio /etc/skel se copia al directorio de conexión del usuario en el momento de la creación

- Cuando se da de alta no se crea el directorio de conexión ni tampoco se crean los permisos o propiedad y se debe copiar el profile por defecto.

Si se quiere que se realice lo anterior automaticamente

#useradd −m usuario

Opciones del useradd:

−u UID Define el UID de usuario.

−o Utilizar esta opción con −u para utilizar UID no unicos en el sistema.

−g grupo Define un ID de grupo o asigna uno existente.

−d dir Directorio de conexión del usuario.

−s shell Del de arranque.

−e expira Fecha en que una presentación expira.

−f inactivo Establece el número de diás que la presentación puede estar inactiva.

Supresión de un usuario.

#userdel −r usuario.

−r Borra el directorio.

Adición de un grupo.

- Creación de un grupo puede ser util en casos en que existan un gran nuemro de usuarios que tengan permisos para un conjunto particular de archivos o directorios.

Para añadir un grupo :

#groupadd

La orden añadirá una linea al /etc/group y le asignará un ID de grupo.

Supresión de un grupo.

#grupodel nombre_grupo

Preparación de terminales e Impresoras.

Es necesario identificar en el sistema UNIX los terminales, impresoras u otrs perifericos harware conectados a la computadora.

Puertos Cada puesto hardware fisico esta representado por un archivo en el directorio /dev. Es a traves de este archivo denominado archivo especial de dispositivo es desde el que se grestiona el dispositivo hardware.

- Una vez conectado el hardware es necesario configurarlo en el sistema operativo. (Conveniente hacerlo mediante sysadm) aunque esposible hacerlo manualmente.

Añadir una impresora en el sistema.

Cuando estan instaladas las utilidades de impresora de linea existe generalmente un guion shell encargado de la gestión del planificador de lp . Para instalar una impresora es necesario detener este planificador y posteriormente reiniciarlo.

- Se debe cambiar la propiedad del puerto a la impresora.

#chown lp /dev/term/11

#chgrp bin /dev/term/11

- Asegurarnos de que tiene el usuario los permisos correctos para la impresión.
- Grupo propietario es el /bin
- Y los permisos 600

#chmod 600 /dev/term/11

- Para desactivar el planificador lp:

/usr/sbin/lpshut (Desactiva demonio de impresora).

- Para identificar la impresora en el sistema.

#lpadm −e {nombre de impresora en sist} −i {modelo} −l {terminal asociado}

- Para reiniciar el planificador de la impresora

#lpsched −d (impresora por defecto).

- Para capacitar la impresora para realizar trabajos:

#enable {nombre de impresora}

Sistemas de archivos.

Los sistemas de archivos son areas especificas de los medios de almacenamiento fisico donde se almacena la información. Cuando se monta un sistema de ficheros se hace accesible desde un punto de la estructura de directorios del sistema UNIX.

#mount Visualiza los dispositos montados en el sistema.

Visualizar el espacio en disco Para ver el espacio que queda disponible en cada sistema de ficheros:

#df −t

Visualizar la utilización del disco Si se esta agotando el espacio en disco del sistema se puede visualizar el espacio utilizado por cada directorio del sistema con la orden **du**

#du −s Recursiva.

Bit de permanencia Una de las caracteristicas principales de los permisos para la administración es la posibilidad de establecer el bit de permanencia {Sticky bit} para un archivo ejecutable o para un directorio.

$chmod 1000 archivo

$chomod −t archivo

Para cambiar los permisos del archivo se ha de ser su dueño.

La especificación de este bit tiene dos efectos inportantes dependiendo si es un archivo ejecutable o un directorio.

- Si es un archivo ejecutable el sistema operativo no suprimira el programa de memoria cuando terme la ejecución de este de forma que no necesite ser cargado la siguiente vez que se ejecute.
- Si es un directorio permite evitar que ciertos procesos borren los ficheros de un usuario accidentalmente.

Gestion de copia de seguridad y restauración.

La copia de seguridad es un metodo para realizar copias de archivos desde el medio de almacenamiento permanente del sistema a otro medio

Planes de copia de seguridad y restauración Existen varios palnteamientos con respecto a la realización de copais de seguridad .

El primero es realizar copias ocasionales de los datos contenidos en el sistema por no resultar estos impoortantes.

Las ordenes cpio y las ordenes tar sirven para copiar determinados ficheros con su nombre a un medio de respaldo mientras que volcopy se utiliza para realizar copias de seguridad de un sistema de ficheros completos

El segundo planteameinto consiste en elavorar un okan de cpais de seguridad para la que existe ena nueva facilidad en el sistema V.

Copia de seguiridad y restauración con **cpio.**

Fue creada para sustituir al tar . Su principal ventaja es la flexibilidad. Cpio acepta su entrada de la entrada

estandar y da la salida en la salida estandar.

Asi puede recibir la lista de ficheros de la copia de seguridad mediente el contenido de la orden ls, cat, find, etc.

Pudiendo dirigir la salida a cualquier medio de respaldo del sistema.

Modos de operación con cpio Opera en tres modos :

−o Modod de salida Se le proporciona una lista de ficheros y el destino y empaque dichos ficheros en un unico fichero cpio en el destino.

−i Modo de entrada Se el proporciona un archivo cio y lo divide de nuevo en los archivos y directorios originales.

−p Modo de paso Funciona como modo −o pero en lugar de enpaquetar cada archivo se vincula individualmente a un directorio en el sistema de ficheros del sistema UNIX.

Copia de seguridad a un disquete

Find . * −print | cpio −ocv > /dev/fd0

Restauracion Se colocara en el directorio donde se desea restaurarlos.

Cpio −ivcd < /dev/fd0

Ej en modo de paso

$find . * −orint | cpio −pmvd /mnt/mcnbackup

$cpio

Opciones:

−d Crea directorios cuando se necesitan. Es util cuando se copia un arbol de directorios.

−v Imorime los nombres de los ficheros que estan siendo procesados, con t, proporciona un listado más detallado.

Copia de seguridad con **tar**

El nombre de la orden tar se refiere a un archivos en cinta. Debido a su extenso su antes de la aparición de cpio, ha seguido evolucionando ciomo utilidad de copia de seguiridad y restauración.

Al igual que cpio tra puede hacer copias de seguridad de archivos y directorios individuales sobre difrenctes medios, y posteriormente restaurar estos archivos.

$tar El comando tar graba ficheros en un archivo y recupera ficheros de un archivo. .Si no se da el nombre del archivo se utilizara por defecto un nombre de fichero del sistema.

−r Añade el fichero nombrado al final del archivo.

−x Extrae del archivo copias de los fichero nombrados.

−t Lista los nombres de todos los ficheros de un archivo.

−c Crea un nuevo archivo, empezando la escritura al principio.

−v Activa el modo de comentario extenso.

−w Activa modo de conformación.

UNIX SYSTEM V v.4 Guillermo Mtez.−Iturralde Gómez

GRUPO FM21

Nº EXP= 951115

1

Pág. 56