Introduccion a la shell

- **Objetivo**
  - Aprender los fundamentos del uso de la línea de órdenes de Linux

- **Contenido**
  - Introduciendo el Shell
  - Aprendiendo las órdenes
  - Variables de ambiente. Alias. Historia
  - Ordenes mas usadas y ayuda de ordenes.
  - Redirigiendo los flujos de información
  - Ensamblando tuberías de órdenes

Introduciendo el SHELL

- **SHELL**: es un programa que interactúa vía teclado con el sistema operativo a través de órdenes, es un interprete de ordenes .

- El shell más utilizado en linux es el BASH (Bourne Again Shell), aunque también existen otros tipos de shell tales como el sh, ksh, csh, entre otros.

Características de BASH

- Soporta la introducción de órdenes por parte del usuario
- Provee un poderoso lenguaje de guiones
- Mantiene un registro histórico de órdenes (HISTORY)
- Permite la completación automática de los nombres de archivo
- Soporta el redireccionamiento y las tuberías

Inicio de el SHELL

- Para iniciar una sesion de bash, debe ingresar a una consola de terminal y aparece un promt de ordenes.

- Todas las ordenes finalizan con un carrier return (<cr>).

- El shell responde a un <cr> interpretando la linea y una vez que esta es completada el shell intenta identificar la orden a ser ejecutada.

- Las ordenes tienen la siguiente sintaxis en forma general:

  - Orden [Opciones] [Argumentos]

    - Opciones: son los modificadores del comportamiento de la orden.

    - Argumentos: son los paràmetros que se le pasan a la orden.

Comodines de el SHELL

Los comodines sustituyen la lista de archivos que se pasan como argumentos a las ordenes.

*	Sustituye cero o mas caracteres a partir de la posiciòn que se encuentra.
?	Sustituye un solo carácter a partir de la posiciòn que se encuentra.
[rango]	Sustituye un solo carácter basado en el rango a partir de la posiciòn que se encuentra.
[!rango]	Sustituye un solo carácter excluyendo el rango a partir de la posiciòn que se encuentra.

```
$ ls a* #los que comiencen por a
$ ls a?c #cualquiera en el segundo.
$ ls a[c-e] #segundo entre c y e.
```

Uso de Variables en el SHELL

- El shell utiliza las variables para llevar un control interno de los procesos (entorno) y pueden venir predefinas por el sistema o ser definidas por el usuario.

- Los nombres de las variables pueden consistir de letras, numeros o simbolos, pero no deben comenzar con un numero.

- Use el simbolo “=” para asignar un valor a una variable.

- Use el simbolo “$” para accesar al contenido de la variable.

- Si el valor tiene espacios en blanco debe entrecomillarse.

```
$ VARIABLE_1=curso
$ echo $VARIABLE_1
$ curso
$ VARIABLE_2=”curso de Linux”
```

Uso del entrecomillado en el SHELL

- El shell interpreta los textos entrecomillados de la siguiente manera:
  - Las comillas dobles “ “ expanden los caracteres $, \, * que esten en el texto.
  - Las comillas simples ' ' no expande ningun carácter que este en el texto.
  - Las comillas tildes ` ` ejecutan como una orden lo que este en el texto.

```
$ CURSO=Linux
$ echo “El curso es $CURSO”
$ El curso es Linux
$ echo 'El curso es $CURSO'
$ El curso es $CURSO
$ echo “El directorio es `pwd`”
$ El directorio es /home/curso
```

Variables Predefinidas en el SHELL}

- El shell tiene algunas variables Predefinidas o Reservadas que usualmente son solo lectura (readonly), entre ellas tenemos las siguientes:
  - HOME (Directorio de login del usuario actual).
  - PATH (Rutas donde va a buscar las ordenes ejecutables).
  - PWD (Directorio actual de trabajo).
  - UID (ID del usuario actual).
  - USER (Nombre del usuario actual).
  - SHELL (path absoluto del shell actual).
  - LANG (lenguaje para las aplicaciones).

Historia de Ordenes en el SHELL

- El bash salva las ordenes entradas por el usuario en un archivo (bitacora) que es almacenado de acuerdo a la variable del ambiente HISTFILE.
  - Use la orden **history** para ver la bitacora de ordenes.
  - Use el carácter “!” con un numero o string para ejecutar una orden de la bitacora.
  - La combinacion de las teclas control (ctrl) y la “r” realiza una busqueda en orden reverso por la bitacora.

```
$ history
1 pwd
2 ls
3 echo “Hola”
$ !1
/home/curso
$ !pw
/home/curso
$ !!
/home/curso
```

Alias de Ordenes en el SHELL

- El bash soporta alias de ordenes, usados para crear abreviaciones o nombres alternativos para las ordenes actuales:
  - Use la orden **alias** para ver los alias definidos.
  - Para definir un nuevo alias utilice la siguiente sintaxis general:

    alias *nombre_alias=orden*

  - La orden **unalias *nombre_alias*** elimina un alias previamente definido.

```
$ alias
alias dir='ls -l'
$ alias borrar='rm -i'
$ borrar prueba
Borrar el archivo prueba? S
$
```

Ayudas en el SHELL

- El man (manual electronico), pemite obtener una ayuda detallada y precisa. Esta compuesto de varias secciones de clasificaciòn.

  Sintaxis: **man *[seccion] archivo/orden***

- El info permite obtener una ayuda detallada y precisa, ademas de permitir navegar por topicos seleccionados.

  Sintaxis: **info *[seccion] archivo/orden***

- La opcion --help o -h que viene construida internamente en las ordenes y presenta por lo general una ayuda resumida..

  Sintaxis: **orden *--help***

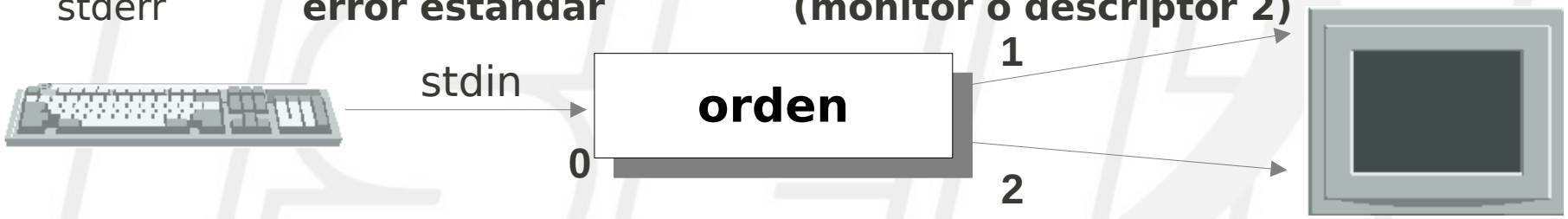
Ordenes para Archivos

- **cat *archivo*** (muestra el contenido sin pausas)
- **more *archivo*** (muestra el contenido con pausas)
- **less *archivo*** (muestra el contenido con avanze y retroceso de página)
- **head *archivo*** (muestra por defecto las primeras diez líneas)
- **tail *archivo*** (muestra por defecto las ultimas diez líneas)
- **wc *archivo*** (muestra la cantidad de lineas, palabras y/o caracteres)
- **cut [opciones] *archivo*** (corta el contenido de un archivo).
- **sort [opciones] *archivo*** (ordena el contenido de un archivo).

Entrada, salida y errores estándares

- **Las órdenes tienen de manera estándar tres flujos de información de entrada/salida:**

stdin	**entrada estándar**	**(teclado o descriptor 0)**
stdout	**salida estándar**	**(monitor o descriptor 1)**
stderr	**error estándar**	**(monitor o descriptor 2)**



- **Los flujos de E/S pueden redirigirse haciaa/desde archivos/dispositivos**
- **Los flujos de E/S tienen números asociados con ellos llamados descriptores:**

Redirigiendo la salida estándar

- Se redirige stdout a un archivo con 1>
  - Si se omite el número antes de >, el valor por defecto es 1
  - El archivo contendrá lo que normalmente aparecería en la pantalla
  - Se sobrescribe el contenido previo del archivo
- Las órdenes no interpretan los datos que siguen luego de la redirección
- El ambiente realiza la redirección de la salida

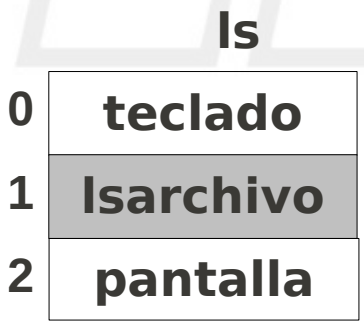
```
$ ls dir1 1>lsarchivo
$ more lsarchivo
archivo1
archivo2
$
```

Le indica al ambiente redirigir la stdout de esta orden al archivo lsarchivo

Los flujos de información de entrada y salida están separados

- La redirección de stdout no afectará a stderr o stdin
- Los mensajes de error se muestran en stderr, quien sigue conectada a la pantalla del terminal

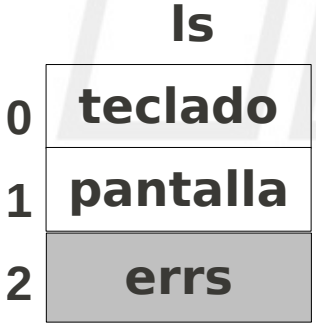
```
$ ls dir2 > lsarchivo
ls: dir2: No such file or directory
$ more lsfile
$
```



En este caso no hubo salida estándar, de modo que lsarchivo **está vacío**

Redirigiendo el error estándar

- **Redirija** stderr **a un archivo con 2>**
- **La redirección de** stderr **no efectará a** stdout **ni a** stdin
- **La salida normal se muestra en** stdout**, quien todavía está conectada a la pantalla del terminal**




```
$ ls dir1 dir2 2>errs
archivo1
archivo2
$ more errs
ls: dir2: No such file or directory
```

Le indica al ambiente redirigir la stderr **de esta orden al archivo** errs

Anexando stdout y stderr

- **La redirección de** stdout **y** stderr **sobrescribe el archivo de destino**

- **Use el operador de anexión** >> **para preservar el contenido de los archivos**

```
$ more hola
Hola, mundo
$ cat hola >bitacora
$ ls dir1 >>bitacora
$ ls dir2 2>> bitacora
$ more bitacora
Hola, mundo
archivo1
archivo2
ls: dir2: No such file or directory
```

Evitando la salida Redireccionada

- La redirección de stdout y stderr sobreescriben por defecto los archivos de salida, si existe el archivo lo reemplaza y si no existe lo crea como nuevo.

- Para evitar este comportamiento use el siguiente comando de la shell:

    set *-o noclobber*

- Para volver al comportamiento original de no protecciòn a sobreescritura utilice :

    set *+o noclobber*

```
$ ls -l > resultados
$ set -o noclobber
$ ls -l /etc >resultados
$ bash: resultados: no se puede sobreescribir un fichero existente
```

Redirigiendo la entrada estándar

- Redirija stdin desde un archivo con <
- Muchas órdenes Linux leen desde stdin por defecto

```
$ write mensaje
Este exto aparecerá en la pantalla
en la cual esté trabajando root
^D
$
$ write root < mensaje
$
```

write

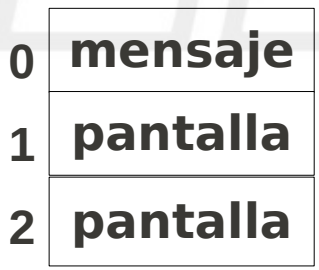
0	**mensaje**
1	**pantalla**
2	**pantalla**

Se redirije stdin **para que se lean datos desde un archivo**

Redirigiendo la entrada estándar

- **Existe un redireccionamiento especial que simula el fin de archivo (EOF) que es un CTRL D desde el teclado.**

```
$ cat >mensaje <<fin
Este texto se almacenara en el
Archivo de mensaje hasta que encuentre
fin
$
```



Al encontrar la etiqueta de fin simulara un EOF o CTRL D para finalizar la entrada estandar

Ejemplos usando redirección

- Use cat con la salida redirijida para crear archivos

```
$ cat > nuevoarchivo
...
^D
$
```

- Use el archivo especial /dev/null para deshacerse de datos indeseados

```
$ ls -l dir1 2>/dev/null
archivo1
archivo2
$
```

```
$ ls -l dir1 1>>mi_lista 2>/dev/null
$
```

Uso de Tuberias (pipes)

- Se pueden conectar ordenes usando las tuberias.

- La salida estandar de una orden (stdout) pasa a ser la entrada estandar (stdin) de la siguiente orden al lado del pipe (|)

```
$ who
Curso pts001   Jul 25   11:01
Root  pts002   Jul 25   11:03
$ who >tmpfile
$ wc -l < tmpfile
   2
$ who | wc -l
   2
```

Tuberias Multiples

- Las tuberias de ordenes pueden colocarse en secuencia logica para formar ordenes compuestas.

- Esto permite hecer operaciones mas complejas desde ordenes muy simples creando los conocidos filtros.

```
$ who | grep ttyp | wc -l
$ who | cut -d ' ' -f1| sort
$ cat /etc/passwd | cut -d : -f 1,5|sort
```