



*Facultad de Ingeniería*  
*Universidad de*  
*Buenos Aires*

75.08 Sistemas Operativos

Lic. Ing. Osvaldo Clúa

Lic. Adrián Muccio

**Shell Scripting**

# Introducción a los Sistemas Operativos

Algunos de los temas que veremos en este curso:

- Scheduling o cómo se decide que será lo próximo que va a ejecutar la CPU
- Administración de Memoria
- Exclusión Mutua, Semáforos
- Máquinas virtuales
- Persistencia de datos y ejecutables
- Device Drivers
- Shell Scripting y Expresiones Regulares

# Unix

¿Qué es Unix?

Evolución desde Multics

Sistemas Abiertos

Sabores

Dennis Ritchie

Ken Thompson



# Unix

## Cultura Unix

Lenguajes: (B->) C, C++, C\*, Java

Shells: sh, ksh, csh, bash

Expresiones Regulares y AWK

Editores: vi / vim

Protocolos: TCP/IP, Ethernet, HTTP, etc

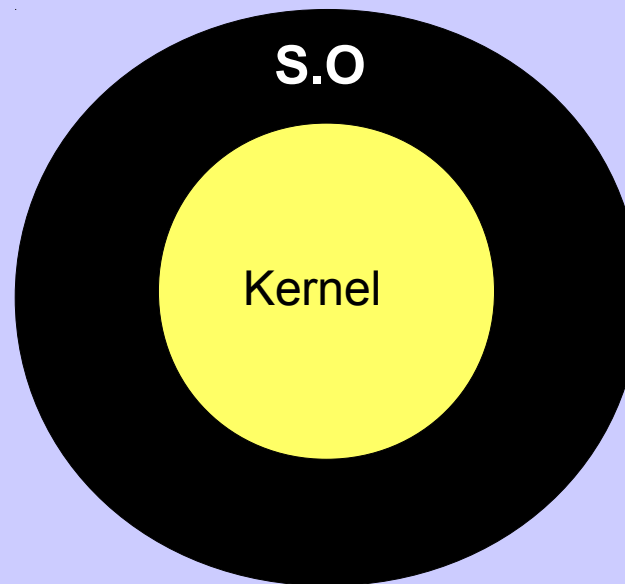
Otras Herramientas: entornos Xwindow, GNU, etc

# Unix

## Características Principales:

Multiproceso / Multiusuario

Los recursos del sistema son administrados por el Kernel



# Unix

El Kernel implementa los servicios esenciales del S.O.:

Administración de Memoria

Administración de Procesos

Concurrencia

# Unix

Todos los procesos se comunican con el Kernel por medio de llamadas al sistema conocidas como *System Calls*

Son un listado de funciones con prototipo estandarizado

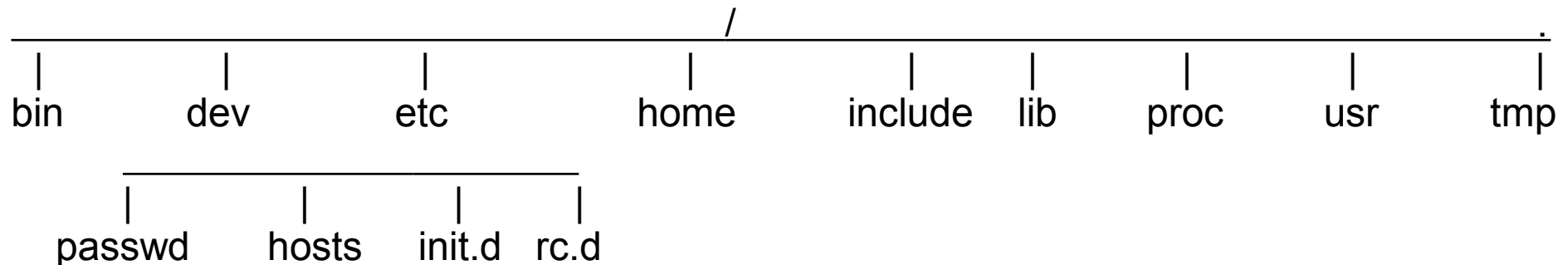
Ejemplo: read, write, exit, etc

# Unix

## File System

Es una estructura jerárquica, agrupada por directorios

## Ejemplo de ordenamiento del File System





# Unix

*“Everything in Unix are Files”* - Kernighan / Pike

Para standardizar y simplificar la forma de acceso a los recursos, Unix los trata como archivos

Los directorios, las terminales, el teclado, los dispositivos de I/O tanto magnéticos como ópticos son archivos

# Unix

Si hacemos `ls -l`, listamos el contenido del directorio con los permisos de cada uno de los archivos, en este caso

Permisos	links	dueño	grupo	tamaño	fecha	nombre
-rw-rw-r--	1	amuccio	grupo1	200	Jul 30 17:41	hola.sh

# Unix

Cada archivo / directorio tiene asociado un conjunto de permisos, los permisos son:

Lectura

Escritura

Ejecución

# Unix

Los archivos se crean con un conjunto de permisos por defecto asignado por el administrador

Se puede modificar mediante el comando umask

```
> umask -S          # lista el modo  
> umask u=rw,g=rw,o=r # setea 664
```

# Unix

En primera instancia el owner es el usuario que crea el archivo, luego se lo puede asignar a otro usuario con el comando *chown dueño archivo*

El grupo es el “principal” del usuario que crea el archivo, se puede cambiar el grupo con el comando *chgrp grupo archivo*

# Unix

Ejemplos dispositivos:

`/dev/rmt/1`: cinta

`/dev/hda5`: disco

`/dev/pts0`: terminal

`/dev/null`: eliminación del sistema

# Unix

Unix maneja los conceptos de *Xterm* y *Terminal Virtual*

*Terminales Virtuales* son terminales de texto que estan implementadas en el mismo host y se accede desde la consola presionando ALT-F2, ALT-F3, etc.

También son terminales virtuales, las que abrimos desde un host remoto, si el host no fuera Unix necesitaremos un emulador de terminal.

# Unix

Unix es *FULL DUPLEX*, los caracteres que se tipean, se envían al sistema (Kernel), el cual los pasa a la terminal

Este es el proceso conocido como *eco*, se puede desactivar por ejemplo para el tipeo de claves



# Unix

Además del eco hay otras propiedades de la terminal que se pueden configurar, por ejemplo:

```
> stty -echo #elimina el eco  
> no se ve lo que escribo  
> stty echo #restaura el eco  
> stty erase backspace > stty erase backspace  
> stty intr ^C # Interrupcion de procesamiento
```

# Shell Scripting

Shell

Interfaz entre SO y usuario

Interpreta comandos

Determina formas de ejecución

Expande caracteres comodines

Expande variables de ambiente

Lenguaje de scripting

# Shell Scripting

## Ingreso al Shell

En una terminal del sistema se autentica el usuario con su clave y en ese momento se le asigna una sesión al usuario

Un usuario puede tener “n” sesiones abiertas en distintas terminales

Una vez que el sistema devuelve el control, aparece la línea de comandos del shell que el administrador del sistema le asignó al usuario

# Shell Scripting

## Ingreso de Comandos

```
> echo "Hola Mundo"  
Hola Mundo  
> _
```

# Shell Scripting

## Algunos Comandos útiles:

adduser	cp	ln	ls	find
cal	mv	pwd	grep	tar
date	rm	wc	sed	type
sort	man	ps	kill	diff
batch	head	set	mkdir	chown
nohup	tail	talk	more	sleep
read	tr	touch	cat	who

# Shell Scripting

## Primer Shell Script

```
> vim hola.sh
```

```
# Mi primer shell  
echo Hola Mundo
```

```
:wq
```

# Shell Scripting

Para poder ejecutar nuestro script

```
> chmod +x hola.sh  
> hola.sh  
Hola Mundo  
> _
```

Otra sintaxis para los mismos permisos

```
> chmod 775 hola.sh  
> hola.sh  
Hola Mundo  
> _
```

# Shell Scripting

## Ejecución *Foreground* con proceso hijo

```
> script1.sh
```

*script1.sh necesita permiso de ejecución  
no nos devuelve el control hasta que no finaliza*

```
> _
```

```
> cp origen.dat destino.dat ; more destino.dat
```



# Shell Scripting

## Ejecución *Background* con proceso hijo

> script1.sh &      *script1.sh necesita permiso de ejecución*  
                         *Nos devuelve el control en el momento*

[1] 20295      *muestra el número de proceso*

> \_

> ps

PID	PPID	TTY	TIME	CMD
5754	1	pts/6	00:00:00	ksh
20295	5754	pts/6	00:00:00	script1.sh
20861	5754	pts/6	00:00:00	ps

>

[1]+ Done      script1.sh      nos avisa que finalizó

# Shell Scripting

## Ejecución *Foreground* sin proceso hijo

```
> . script1.sh    script1.sh no necesita permiso de ejecución  
                  no nos devuelve el control hasta que no finaliza  
                  se ejecuta en el mismo ambiente, eso significa que  
                  no hay un shell hijo  
  
> _
```

# Shell Scripting

El Shell maneja variables de ambiente

No es necesario “definir” una variable, simplemente comienza a existir cuando le asignamos un valor.

Existen variables predefinidas.

**SHELL**    #Nombre del Shell

**PWD**        #Directorio corriente

**PS1**        #Prompt 1

**PATH**        #Directorios donde buscar ejecutables

# Shell Scripting

Tener una lista de directorios donde buscar a archivos para ejecutar es un concepto utilizado en otros S.O. (por ejemplo Windows)

Si el directorio corriente no se encuentra en esa lista, el Shell no va a ejecutar `script1.sh`.

Para poder ejecutarlo desde el prompt, tenemos dos posibilidades:

1. Hacer una referencia explícita al directorio.

> `./script1.sh`      # `.` es el directorio corriente

# Shell Scripting

## 2. Incluir al directorio corriente en el PATH

```
> PATH=$PATH:..
```

Como la variables es del ambiente, cuando nos reconectemos podría ser necesario repetir esta acción.

# Shell Scripting

Para personalizar el ambiente y evitar acciones repetitivas como setear variables de ambiente, umask y otros existen los archivos `.profile` y `.bash_profile`.

Se encuentran en el `$HOME` del usuario y contienen el seteo de acciones a ejecutar por el shell al momento de conexión.

**IMPORTANTE:** Es conveniente testear cualquier modificación a estos archivos manteniendo una conexión abierta durante el test.

# Shell Scripting

## Relación entre *Variable* y *Ambiente*

Supongamos que en un shell script se realiza una asignación.

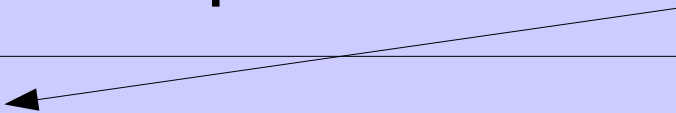
```
# script1.sh  
echo "VARIABLE: $VARIABLE"
```

# Shell Scripting

Ejecutamos script1.sh

```
> script1.sh  
VARIABLE:  
> _
```

VARIABLE sin inicializar



Asignamos valor a VARIABLE y Ejecutamos

```
> VARIABLE="CERO"  
> script1.sh  
VARIABLE:  
> _
```

*Ambientes distintos*





# Shell Scripting

## Utilizamos el comando export

```
> VARIABLE="CERO"  
> export VARIABLE  
> script1.sh
```

```
VARIABLE: CERO  
> _
```

# Shell Scripting

¿Qué sucedería si asignáramos un valor dentro de script?

```
# script1.sh  
export VARIABLE="UNO"  
echo "VARIABLE: $VARIABLE"
```

```
> VARIABLE="CERO"  
> export VARIABLE  
> script1.sh
```

```
VARIABLE: UNO  
> _
```

# Shell Scripting

¿Con qué valor queda VARIABLE?

```
> echo $VARIABLE
```

No se modifica el valor en el padre

```
VARIABLE: CERO
```

```
> _
```

# Shell Scripting

Si ejecutáramos en el mismo *Ambiente*

```
> . script1.sh  
VARIABLE: UNO  
> echo $VARIABLE
```

```
VARIABLE: UNO  
> _
```