



*Facultad de Ingeniería
Universidad de Buenos
Aires*

75.08 Sistemas Operativos
Lic. Ing. Osvaldo Clúa
Lic. Adrián Muccio

Shell Scripting I

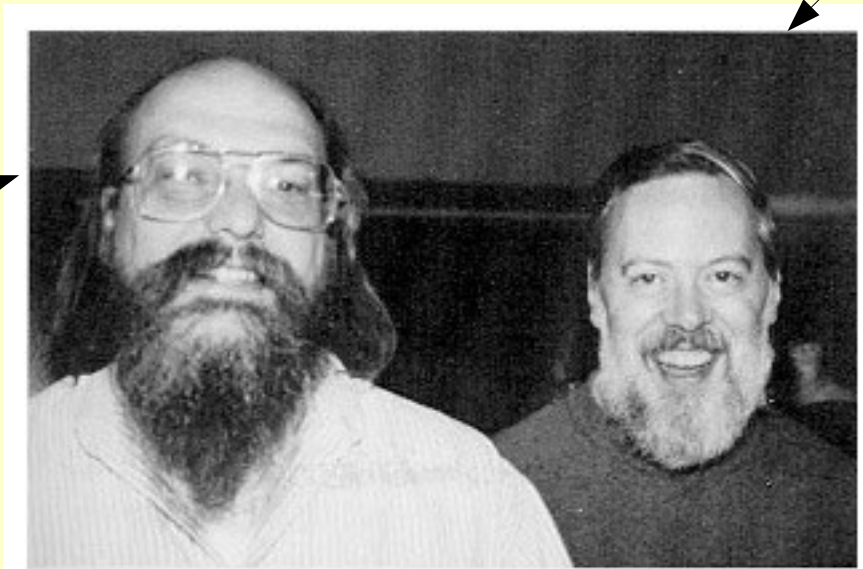
Unix

¿Qué es Unix?

- Evolución desde Multics
- Sistemas Abiertos
- Sabores

Dennis Ritchie

Ken Thompson



Unix

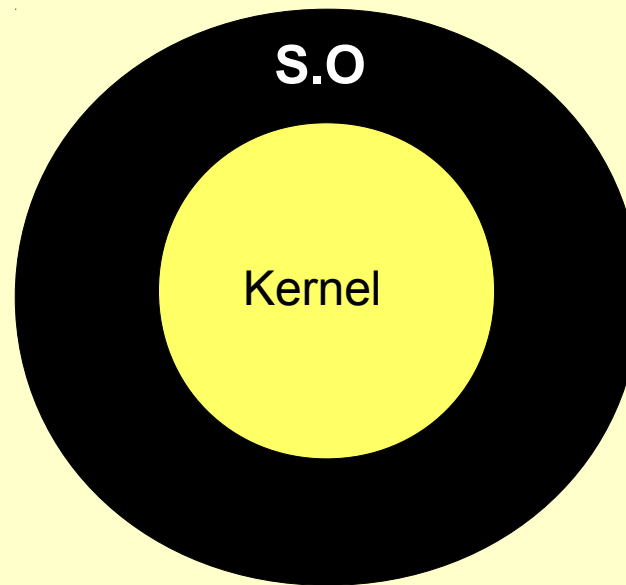
Cultura Unix

- Languages: (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*

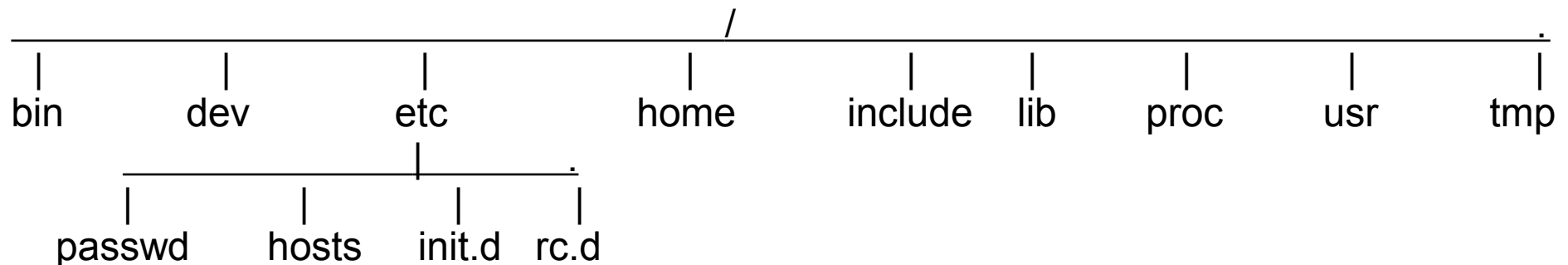
Son un listado de funciones con prototipo standarizado

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

- 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 # Caracter para borrar
> stty intr ^C # Interrupcion de procesamiento
```

Unix

Shell

- Interfaz entre SO y usuario
 - Interpreta comandos
 - Determina formas de ejecución
 - Expande caracteres comodines
 - Expande variables de ambiente
- Lenguaje de scripting

Unix

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 asigno al usuario

Unix

Ingreso de Comandos

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

Unix

Algunos Comandos útiles:

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

Unix

Primer Shell Script

```
> vim hola.sh
```

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

```
:wq
```

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

- 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

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  
> _
```

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

- 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
```

Unix

- 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
```


Unix

- 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  
  
> _
```

Unix

- El Shell maneja variables
- No es necesario “definir” una variable, simplemente comienza a existir cuando le asignamos un valor.
- Existen variables predefinidas.
 - PATH
 - SHELL
 - PWD
 - PS1

Unix

- Relación entre *Variable* y *Ambiente*

Supongamos que en un shell script se realiza una asignación.

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

Unix

- Ejecutamos script1.sh

VARIABLE sin inicializar

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

- Asignamos valor a VARIABLE y Ejecutamos

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

Ambientes distintos

Unix

- Utilizamos el comando export

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

```
VARIABLE: CERO  
> _
```

Unix

- ¿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
> _
```

Unix

- ¿Con qué valor queda VARIABLE?

```
> echo $VARIABLE
```

No se modifica el valor en el padre

```
VARIABLE: CERO
```

```
> _
```

Unix

- Si ejecutáramos en el mismo *Ambiente*

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

```
VARIABLE: UNO  
> _
```