Shell

En la versión 6 de Unix, o tal como se conoce en UNIX v6, Ken Thompson (de Bell Labs) desarrollo en primer shell para UNIX llamado el shell V6 en 1971. Al igual que en Multics el mismo era un programa de usuario independeinte (/bin/sh)que se ejecutaba fuera del nucleo del sistema operativo. Este shell es conocido como osh. El programa estaba escrito en unas 900 lineas de código C.

El Shell introdujo una serie de bloques de construscción que hacen a la filosofia unix:

- Redireccionamiento (> y >>).
- piping (| or ^).
- Ejecución de secuencia; de comandos (con ;).
- Ejecución asincrónica de comandos (con&).

Un problema que tenia el shell de Thompson era la imposibilidad de generar scripts.

Los Shell Modernos

Bourne Shell

El **Bourne shell**, fue creado por Stephen Bourne en AT&T Bell Labs para la versión V& de UNIX, el mismo puede encpntrarse en nuestros dias aun en distribuciones de unix, es más, en muchas distribuciones, el shell del usuario root es el Bourne shell. Este se baso en ALGOL68 por ende la sintaxis es similar.

El Bourne shell tenia básicamente dos grandes propositos:

- Ser un interprete interactivo de ejecución de comandos para el systema operativo
- Poder correr scripts: un script es un programa que puede ser ejecutado por el shell del sistema operativo.

Ademas Bourne shell ofrecia muchas otras caracteristicas que el shell de Thompson no tenia:

- Estructuras de Control
- Loops
- Variables en los scripts

Este shell fue el padre de todos los shells modernos como:

- **Korn Shell** (ksh)
- **Almquist shell** (ash)
- **Bourne Again Shell** (bash)

La perspectiva del usuario :

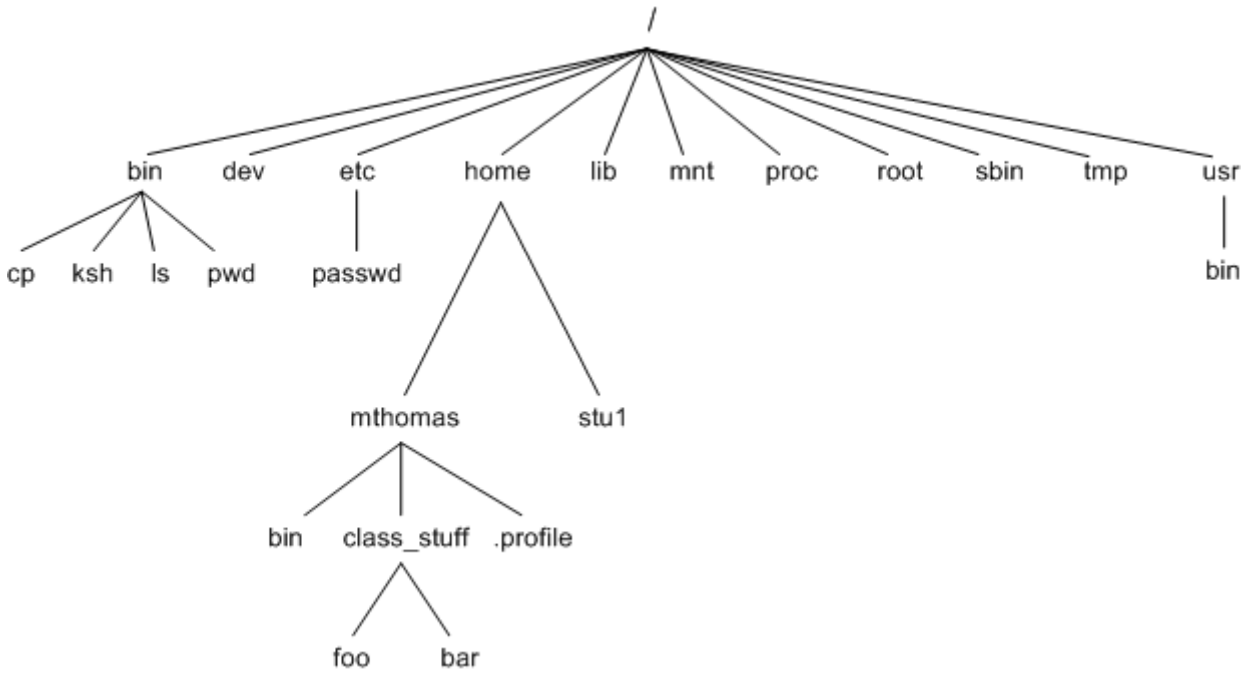
Desde el punto de vista del usuario lo que “se ve” del sistema operativo es:

1. El **Sistema de Archivos** o File System.
2. El **Entorno de Procesamiento**.
3. Los **Bloques Primitivos de Construcción** .

El Sistema de Archivos

El **sistema de archivos** de Unix se caracteriza por:

- su **estructura jerarquica**.
- tratamiento consistente archivos de datos.
- la habilidad de **crear y borrar archivos**.
- el **crecimiento dinámico** de los archivos.
- la **proteccion de los archivos** de datos.
- el **tratamiento de los dispositivos perifericos como si fueran archivos**.



- El Sistema de archivos de unix está organizado como un árbol con una única raiz (root) (que se escribe como “/”); cada nodo no hoja del sistema de archivo se denomina directorio de archivos, y las hojas del árbol pueden ser a su vez pueden ser archivos, directorios de archivos, archivos especiales, etc.

- El nombre de un archivo está dado por el path (camino) para localizarlo dentro de la estructura del sistema de archivos. “/etc/passwd”
- El sistema operativo no tiene noción de cómo es la estructura interna en el que se guardan.
- El sistema operativo Unix provee una lista de programas que vienen junto al sistema operativo, estos programas normalmente se denominan **comandos**, para realizar operaciones sobre los componentes del sistema de archivos.

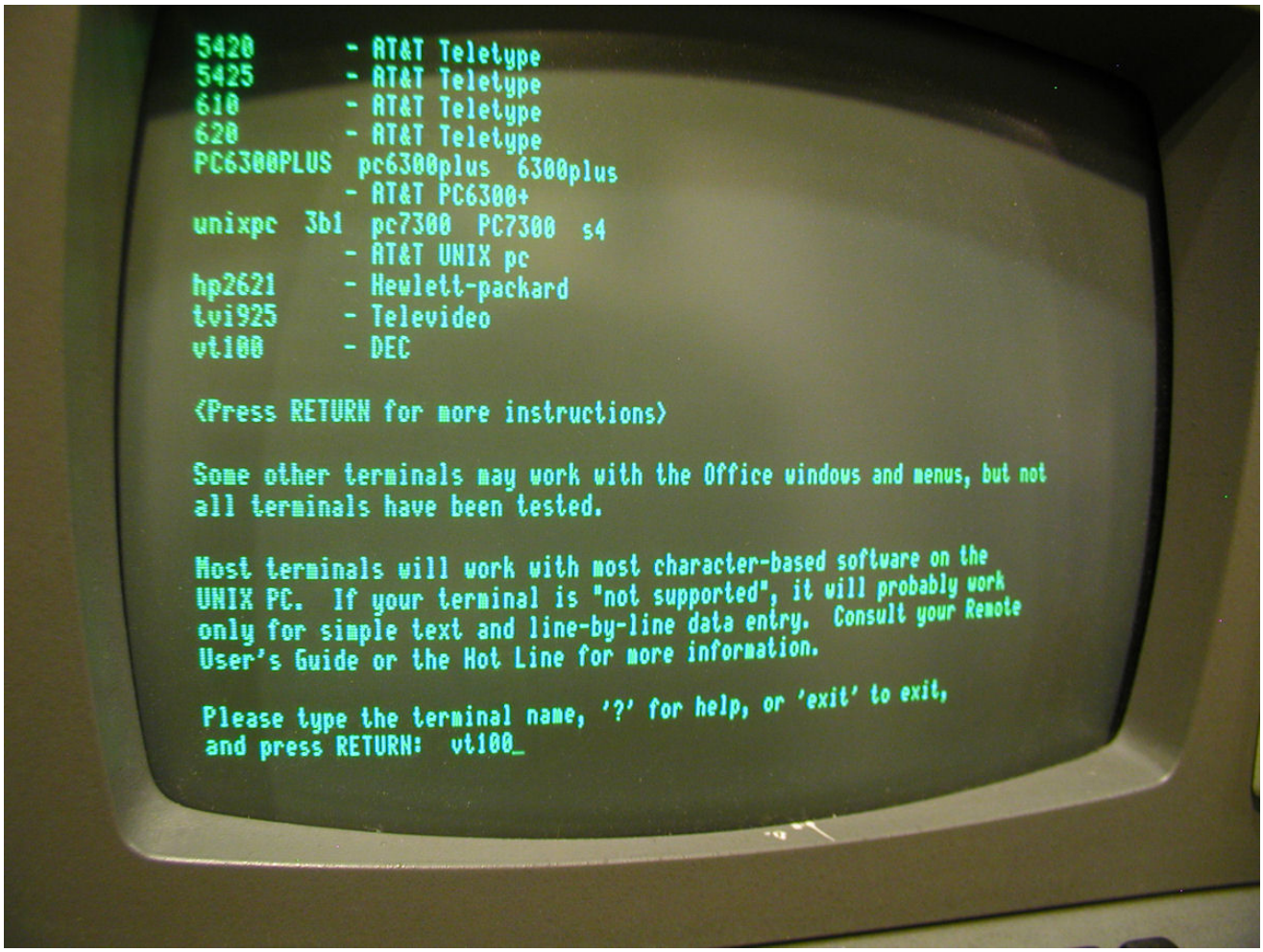
Existe una lista bastante extensa de comandos unix, aquí se deja una lista de los comandos que tienen que estar en una distribución de UNIX Unix Commands

El Entorno de Procesamiento

Un programa es un archivo ejecutable un proceso es **una instancia** de un programa en ejecución.

- Muchos procesos pueden estar ejecutandose a la vez, no existe un límite logico para la cantidad de estos.
- Existen varios comandos relacionados con los procesos.
- El *interprete de comandos* o *shell* es el primer programa que se ejecuta (en un unix) en modo usuario, este a su vez ejecuta el comando de login. El interprete de comandos captura las teclas que son presionadas en el teclado y las traduce a nombres de comandos.
- **El shell puede interpretar tres tipos de comandos:**

    1. **Comandos Ejecutables** simples proramas , por ejemplo el **ls**
    2. **Shell scripts** que es un archivo ejecutable, consistente en lineas de comandos ejecutables
    3. **Estructuras de control del Shell** *if-then-else-fi*



- El shell ejecuta los comandos buscando en ciertos directorios en una determinada secuencia.

Los Bloques de Construcción Básicos

la filosofia de la linea de comandos de UNIX/LINUX se basa en la combinacion de pequeños comandos que se especializan en realizar una única tare, y a partir de los bloques básicos de construcción (**building blocks**) permiten realizar tareas complejas. A continuación se explicaran estos bloques básicos de construcción.

Ejecución Segundo Plano

```
cmd &
```

Ejecuta un comando en segundo plano, es decir la terminal no se congela hasta que el comando termine.

Ejecución en secuencia

```
cmd1;cmd2
```

Ejecuta comandos en secuencia

```
$ cd;ls
```

Ejecución en Pipeline

La ejecución en pipe line es uno de los bloques de construcción del shell más importantes ya que permite conectar una serie de comandos unos con otros. Cuando un programa comienza a ejecutarse siempre hay tres archivos que se abren automaticamente y poseen un descriptor especifico. Estos archivos se llaman los archivos de entradas y salida estándar o por defecto:

Descriptor	Nombre	Abrev.	Dispositivo
0	Estandar Input	stdin	teclado
1	Estandar Output	stdout	pantalla
2	Estandar Error	stderr	pantalla

```
cmd1 | cmd2
```

Ejecuta los comandos en modo pipeline, en el cual la salida del primer comando es la entrada del segundo comando.

Redireccionamiento de Entrada y Salida

Envia la salida estandard del comando al archivo de nombre file (sobre escribe). .. code-block:: console

    cmd > file

Envia la salida estandard del comando al archivo de nombre file en modo append. .. code-block:: console

    cmd >> file

Toma la entrada estandard del comando al archivo de nombre file. .. code-block:: console

    cmd < file

Toma un heredocument .. code-block:: console

    cmd << file

también conocido como el here-document, aqui se muestra un ejemplo:

```
$ wc << here
> esto es un heredocument
> recordar que debe empezar y terminar
> con el mismo tecxto
> here
3 14 81
```

Variables

El shell permite crear variables y asignarles un valor utilizando el operador =. .. code-block:: bash

    $ mi_variable=hola

se accede al valor almacenado de la variable con el signo $ delante del nombre

Built-In

Los built-Ins no son comandos propiamentedichos, sino que son acciones que pertenecen al shell, muchas veces se confunden con comandos:

- **pwd**: Print working directory, permite saber en que parte del sistema de archivos se está.
- **cd**: Chande directory, permite modificar el directorio de trabajo, en el cual se estan ejecutando los comandos.
- **history**: permite ver los últimos comandos ejecutados.
- **exit**: permite salir del shell.
- **echo**: escribe por la salida estandar

Los Comandos

Comandos Varios

- **man**: este comando permite leer las páginas del manual de
- **less**: impresión
- **grep**: busca un texto dentro de uno o varios archivos
- **uname**: lista información del sistema
- **which**:

Comandos de Archivos:

- **touch**: este comando permite crear un archivo o marcarlo como modificado.
- **rm**: este comando permite borrar un archivo.
- **mkdir**: este comando permite crear un directorio.

- **rmdir**: este comando permite eliminar un directorio.
- **ls**: este programa permite listar el contenido de un directorio.
- **cat**: este comando permite ver el contenido de un archivo.
- **cp**: copia un archivo.
- **mv**: cambia la locación de un archivo.
- **find**: busca un determinado archivo en la estructura del filesystem
- **tar**: (t)ape (ar)chive
- **df**: disk free
- **du**: disk used
- **shred**: elimina sobreescribiendo un archivo
- **head**: muestra las 10 primeras lineas de un archivo
- **tail**: muestra las 10 últimas lineas de un archivo
- **diff**: el comando compara dos archivos

Comandos de Procesos

- **ps**: muestra los procesos que ha lanzado un determinado usuario.
- **top**: muestra la lista de comandos que utilizan mas recursos.
- **who**: muestra los usuarios conectados.
- **kill**: mata a un proceso.
- **free**: muestra la memoria libre del sistema.

Comandos de Usuarios

- **who**: lista los usuarios conectados al systema
- **su**: switch user

Arquitectura de un Shell

La arquitectura fundamental de un shell es estructuralmente sencilla. La arquitectura básica es similar a tubería en la cual los datos ingresados por el teclado son en primer lugar analizados y parseados, los simbolos son expandidos y finalmente el/ los comandos son ejecutados.

 images/shell/figure2.gif

TODO: