PROGRAMACIÓN DE SISTEMAS

Unidad 1 – Programación en el Shell

PRÓLOGO

¿Qué es la Programación de Sistemas?

- Es las programación a bajo nivel, para la creación de software del sistema.
- Ejemplos de software del sistema:
  - *Compiladores*
  - *Bases de datos*
  - *Servidores web, etc.*
- Aplicaciones GUI viven en un nivel más alto.

¿Por qué Programación de Sistemas?

- El software de sistema constituyen la infraestructura sobre las cuales corren las aplicaciones de más alto nivel.
- Por ejemplo, si usamos Java, alguien tuvo que implementar la JVM.
- Antes de usar bases de datos, alguien tuvo que implementar el motor de base de datos.

¿Por qué Programación de Sistemas?

- En este curso, haremos programación de sistemas en Linux.
- Nuestra lenguaje de implementación será **C**.
- Los programadores de sistema hacen uso de las **llamadas al sistema.**

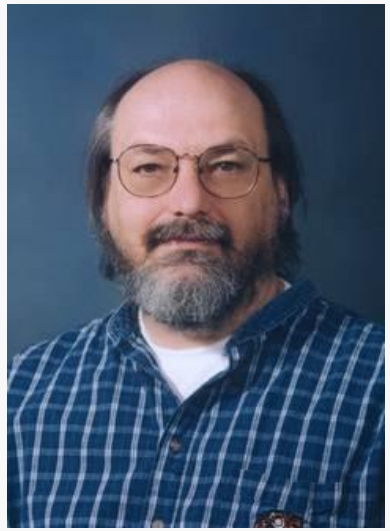
1.1 INTRODUCCIÓN A LA ARQUITECTURA DE LINUX

Introducción a la Arquitectura de Linux

- ¿Que es UNIX?

  UNIX fue originalmente concebido como una "mesa de trabajo" para desarrollo de aplicaciones dentro de Bell Labs en los 70s. Terminó convirtiendose en SO.

- Creado por:



Ken Thompson

Dennis Ritchie (R.I.P.)

Introducción a la Aquitectura de Linux

- UNIX introdujo conceptos importantes:
  - *Sistemas de archivos jerárquico*
  - *Procesos*
  - *Archivos de dispositivos*
  - *Intérprete de línea de comandos*
  - *Redirección de entrada/salida*
  - *Time-sharing*
  - *Multiprogramación*

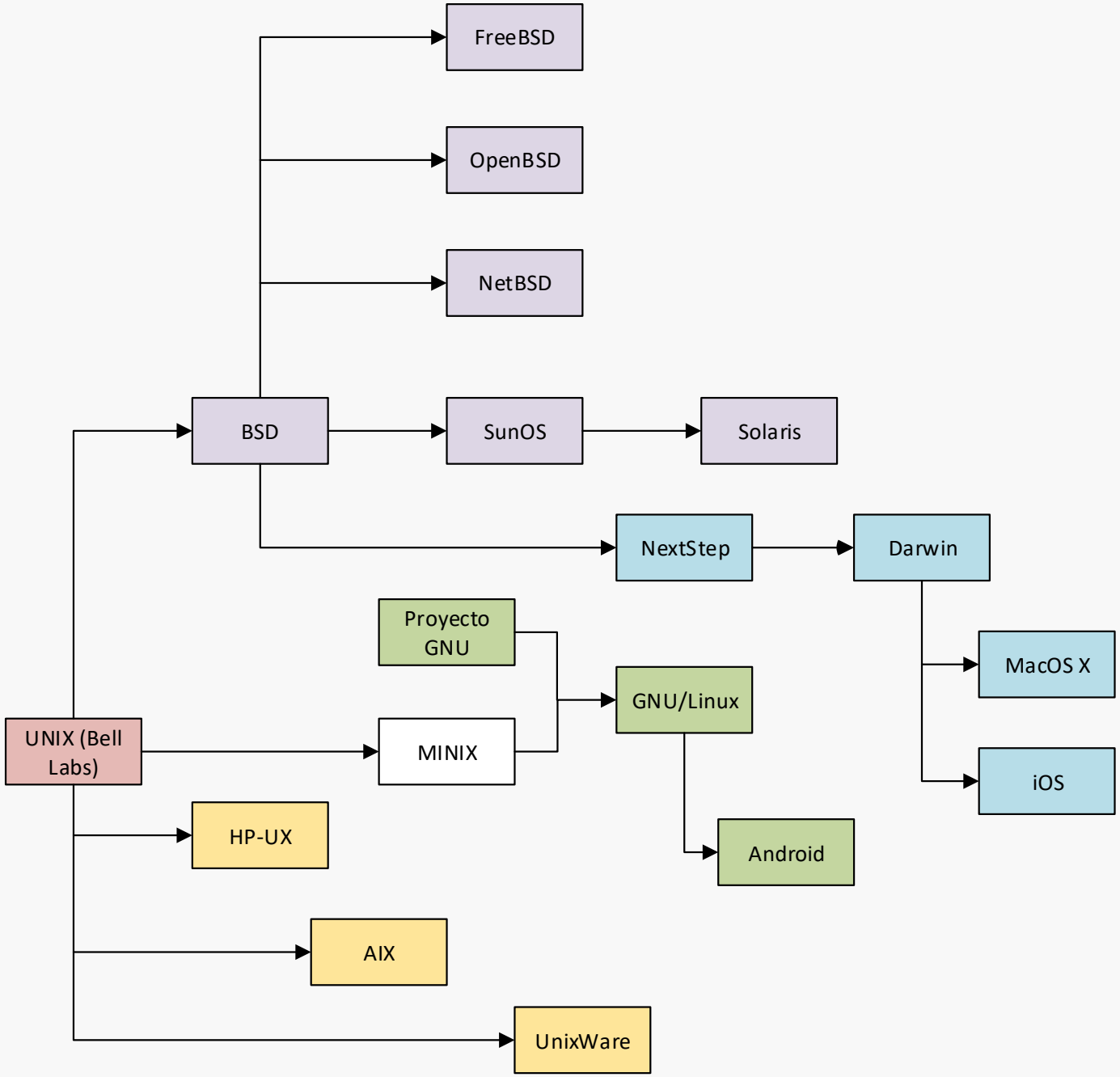
Introducción a la Aquitectura de Linux

PDP-7

PDP-11

Introducción a la Aquitectura de Linux

- UNIX fue escrito ensamblador y en C (también creado por Thompson y Ritchie)
  - *Por lo tanto existe una **relación estrecha** entre C y UNIX.*
- UNIX está estandarizado en la Single UNIX Specification v4 (POSIX:2008).


FreeBSD
OpenBSD
NetBSD
BSD
SunOS
Solaris
NextStep
Darwin
Proyecto GNU
MacOS X
GNU/Linux
UNIX (Bell Labs)
MINIX
iOS
HP-UX
Android
AIX
UnixWare

Introducción a la Arquitectura de Linux

- ¿Qué es Linux?
  - *Linux es un sistema operativo similar a UNIX* ***(Unix-like)***
  - *El componente principal de Linux es el núcleo (kernel), desarollado por Linus Torvalds en 1991.*

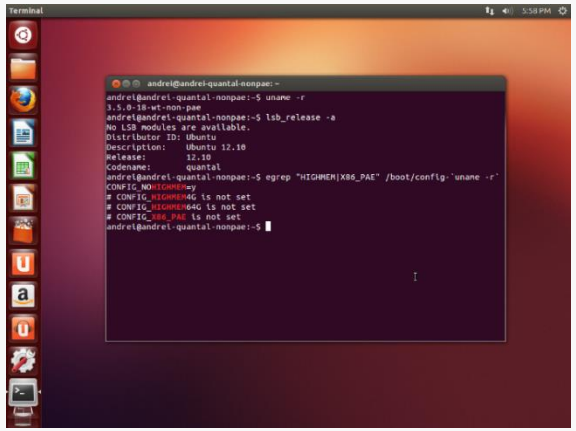
- **Kernel:** el principal componente del sistema operativo , que maneja los aspectos más básicos del hardware y da servicios a las aplicaciones.

Introducción a la Arquitectura de Linux

- Un Sistema *Unix-like* implementa la interface POSIX (Portable Operating System Interface).
- POSIX define interfaces de programación del sistema operativo tales como:
  - *Interfaces al Sistema*
  - *Definiciones base*
  - *Comandos y utilidades*
  - *http://pubs.opengroup.org/onlinepubs/9699919799/*

¿Dónde encontramos a Linux?

Smartphones



PCs

Centros de computo/servidores

Sistemas embebidos

Aplicaciones en tiempo real

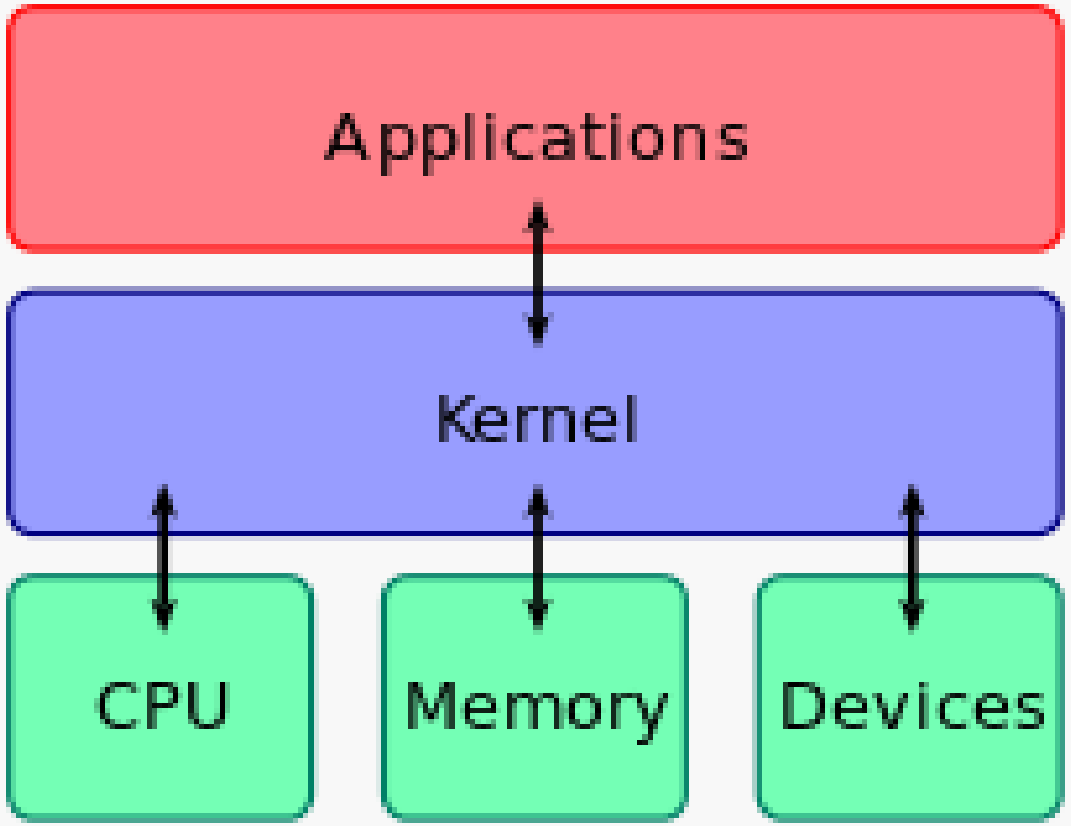
Introducción a la Arquitectura de Linux

- **Estándarización**
  - ***POSIX:*** *Portable Operating System Interface, creada por la IEEE.*
  - *El estándar 1003.1 especifica una interfaz (API), no implementación*
  - *Define los servicios que el SO debe proveer para ser considerado compatible con POSIX*
  - *¡Windows no es POSIX compliant!*
  - *Define los archivos cabeceras (.h) necesarios.*

Introducción a la Arquitectura de Linux

- **Conceptos Generales**
  - *UNIX/Linux es un sistema operativo. Como tal, cumple dos funciones básicas:*
    - Proveer una **abstracción** del hardware.
    - Proveer servicios a los programas que corren sobre él.

- **Kernel**: parte fundamental del sistema operativo que controla los recursos de hardware.
- **Llamadas al sistema:** interface de programación para acceder a los servicios ofrecidos por el *kernel*.


Applications
Kernel
CPU
Memory
Devices

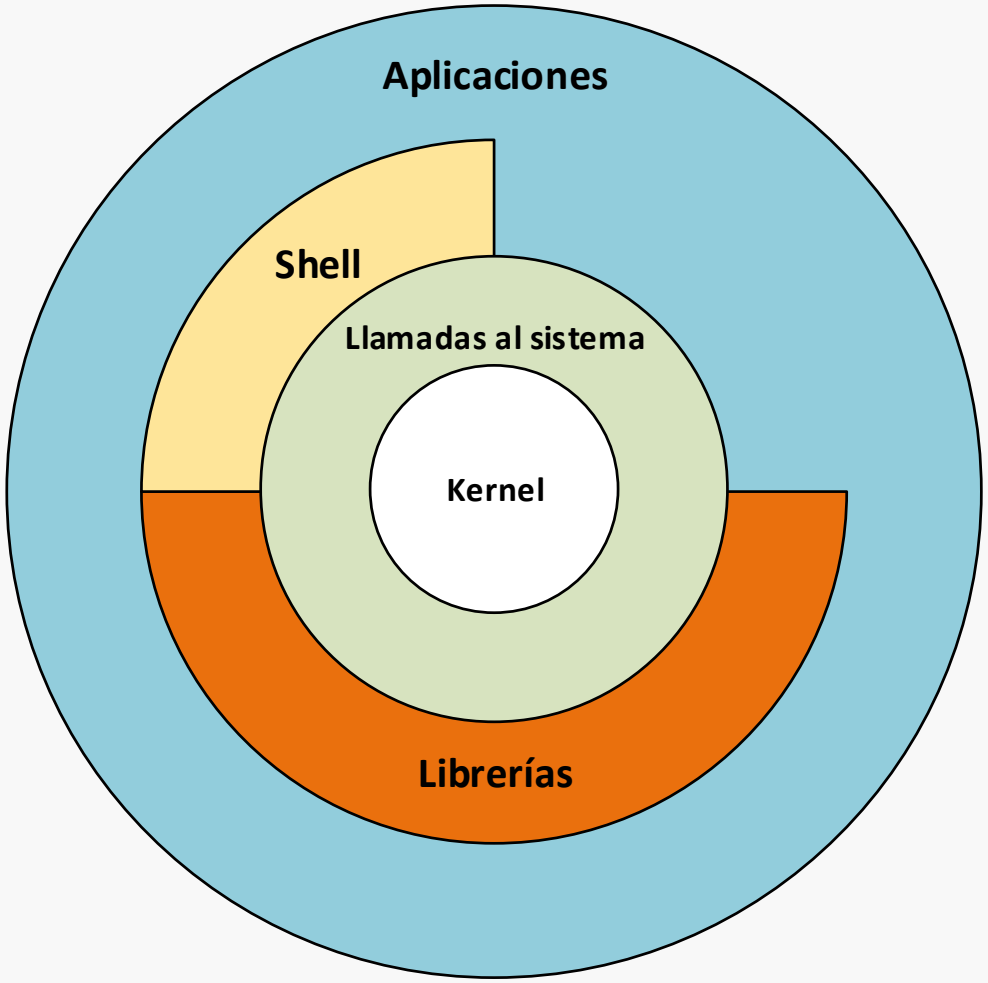
Introducción a la Arquitectura de Linux

- **Abstracción**

Nivel establecido de complejidad de interacción entre sistemas; los **detalles** de la implementación se manejan en un nivel inferior.

Hardware	Abstracción
CPU	Hilos
Memoria	Proceso
Disco	Archivos
Red	Comunicacion (sockets)
Monitor	Gráficos, ventanas, salida
Teclado	Entrada
Mouse	Puntero

Introducción a la Arquitectura de Linux



Introducción a la Arquitectura de Linux

- Al conectarnos a un sistema Linux, debemos proveer usuario y contraseña.
  - *El sistema verificara el archivo* ***`/etc/passwd`*** *(en la actualidad se usan un archivo encriptado).*
  - *Este archivo contiene líneas como:*

```
sar:x: 205: 105: Stephen Rago:/home/sar:/bin/ksh
```

  - *Siete columans: usuario, contraseña encriptada, ID de usuario (205), ID de grupo (105), campo comentario, directorio home, programa shell*

Introducción a la Arquitectura de Linux

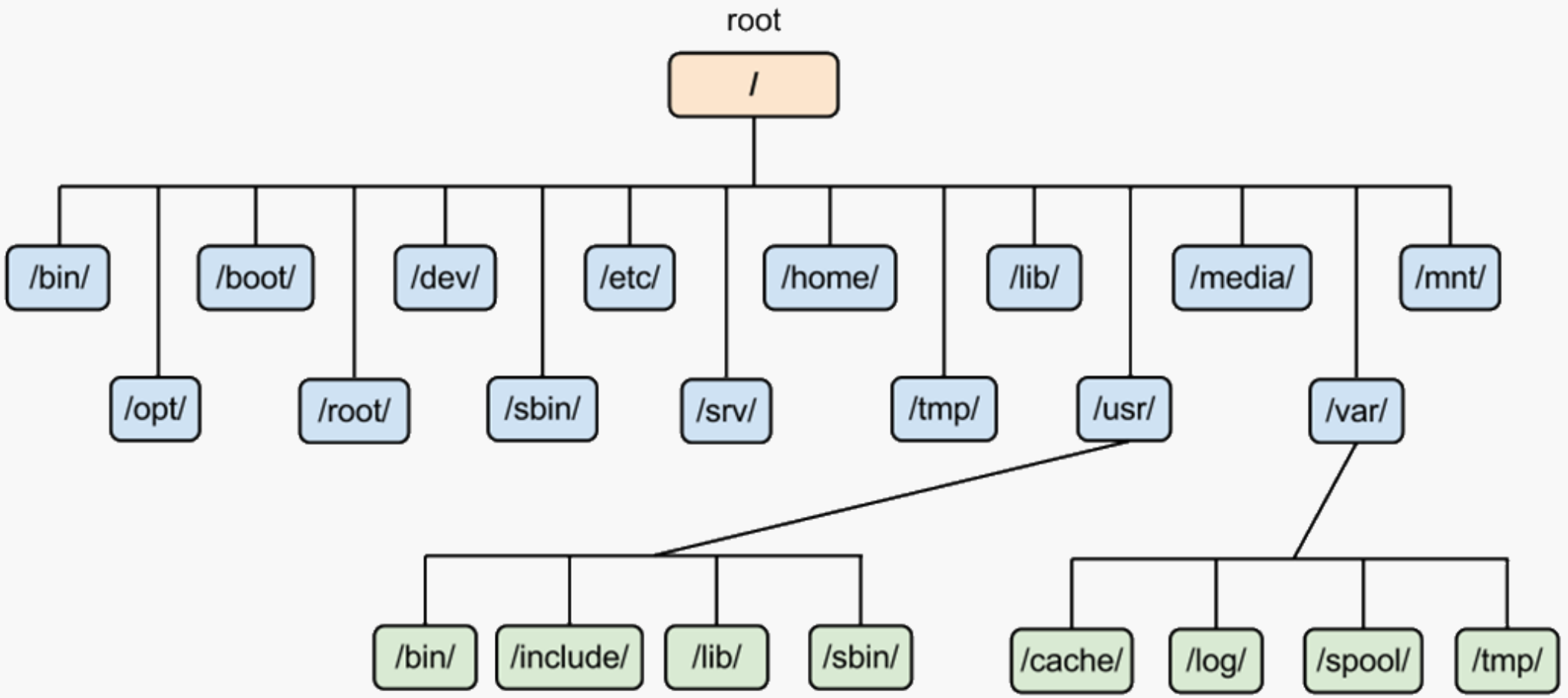
- **Shell:** intérprete de línea de comandos que lee la entrada del usuario y ejecuta comandos.

Shell	Ruta	FreeBSD	Linux 3.2	Mac OS X	Solaris 10
Bourne Shell	/bin/sh	si	si	copia de bash	si
Bourne-again Shell	/bin/bash	opcional	si	si	si
C Shell	/bin/csh	link a tcsh	opcional	link a tcsh	si
Korn Shell	/bin/ksh	opcional	opcional	si	si
TENEX C Shell	/bin/tcsh	si	opcional	si	si

Introducción a la Arquitectura de Linux

Sistema de Archivos

- Sistema de archivos de UNIX/Linux es jerárquico (árbol)
  - *Todo empieza en el directorio root (/)*
- **Directorio:** archivo que contiene entradas de directorio
- **Entrada de directorio:** nombre de archivo + numero de i-nodo


root
/
/bin/
/boot/
/dev/
/etc/
/home/
/lib/
/media/
/mnt/
/opt/
/root/
/sbin/
/srv/
/tmp/
/usr/
/var/
/bin/
/include/
/lib/
/sbin/
/cache/
/log/
/spool/
/tmp/

Introducción a la Arquitectura de Linux

- Sistema de archivos virtual (VFS): Es una abstracción del sistema de archivos actual (AFS)
  - *Provee una interfaz entre el AFS y las aplicaciones*
- Sistema de archivos actual (AFS): Implementación real del sistema de archivos.
  - *Ext4*
  - *NFS*
  - *LogFS*
  - *ReiserFS*
  - *ZFS*

Introducción a la Arquitectura de Linux

- **Nombres de archivo**: el caracter ‘/’ ni el character nulo pueden ser parte del nombre del archivo.
  - *¿Por qué?*
- Dos archivos son creados al crear un directorio:
  - *El archivo . → apunta al propio directorio*
  - *El archivo .. → apuntar al directorio padre*

- ¿A dónde apunta . y .. en el directorio root (/)?

Introducción a la Arquitectura de Linux

- **Ruta de archivos:** secuencia de nombre de archivos separados por slashes

`/home/root/ntpdate`

  - *Ruta absoluta: relativa al directorio root (/)*
  - *Ruta relativa: ruta relativa al **directorio de trabajo**.*

- **Directorio de trabajo**: directorio desde el cual se resuelven las rutas

Introducción a la Arquitectura de Linux

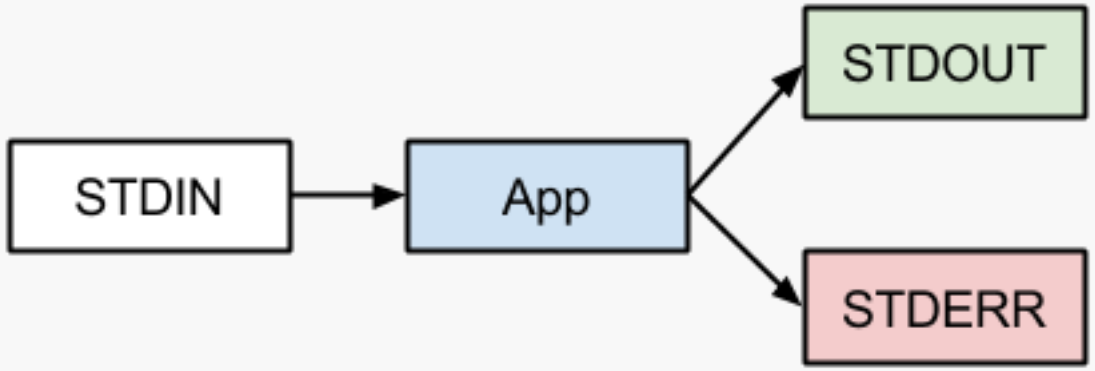
Entrada/Salida

- Para leer/escribir cosas en Linux, lo hacemos a través de descriptrores de archivo.
- **Descriptores de archivos:** Numero entero no negativo para referirse a archivo abierto.

Introducción a la Arquitectura de Linux

Entrada, salida y error estándar.

- `stdin` → *descriptor 0 (usualmente el teclado)*
- `stdout` → *descriptor 1 (usualmente la terminal/shell)*
- `stderr` → *descriptor 2 (usualmente la terminal/shell)*



Introducción a la Arquitectura de Linux

Permisos

- Cada usuario tiene un ID (user ID)
- Usuarios se agrupan en grupos con ID (group ID)

- El *user ID y group ID* nos ayudará a saber que acciones son permitidas en que archivos/Directorios.

Introducción a la Arquitectura de Linux

- Todo archivo/directorio tiene 9 bits que nos dicen los permisos:

`rw--wxrwx`

- **Rojo → Permisos de dueño**
- **Verde → Permisos del grupo del dueño**
- **Azul → Permisos de otros grupo**
- **r –** permiso lectura, **w –** permiso escritura, **x –** permiso de ejecución.

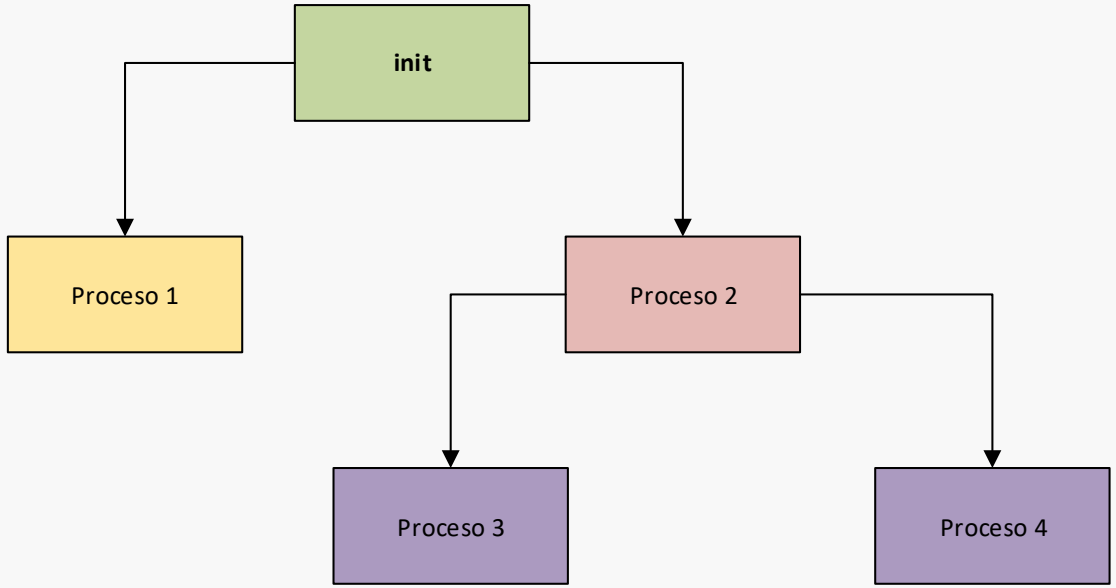
Introducción a la Arquitectura de Linux

Programas y Procesos

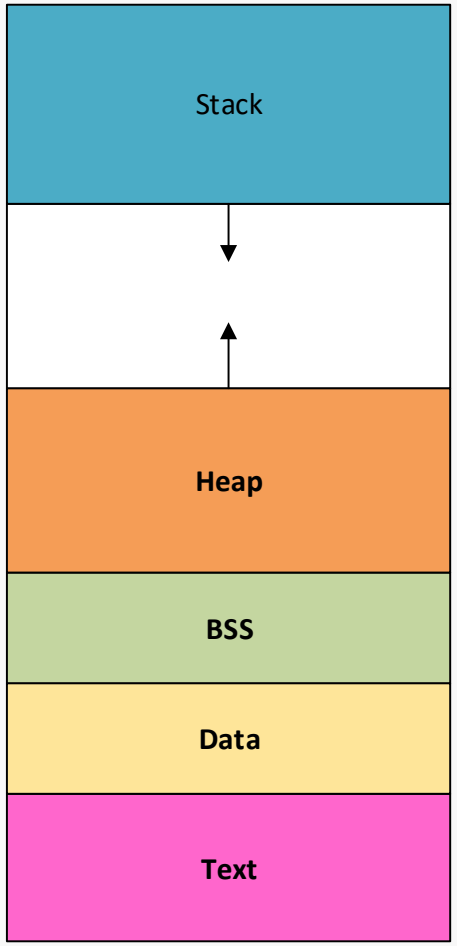
- **Programa**: archivo ejecutable almacenado en disco.
- **Proceso:** instancia de un **programa** en ejecución.
- Todo proceso tiene un identificador único (PID) garantizado por el SO.

Introducción a la Arquitectura de Linux

- Los procesos en Linux siguen una jerarquía de árbol. Todo proceso “nace” de otro proceso.
- El proceso inicial es el proceso *init*



- Espacio de direcciones (address space) de un proceso (memoria virtual).



Introducción a la Arquitectura de Linux

Hilos

- Abstracción del CPU.
- Usualmente un hilo por proceso
- Con múltiplos hilos, podemos hacer mejor uso del CPU (concurrecia).
- Hilos son implementados por la librería *pthreads*

Introducción a la Arquitectura de Linux

Manejo de Errores

- **errno:** variable global donde se guarda el último error.
- En Linux cuando ocurre un error:
  - *Función retorna -1.*
  - *Se establece la variable* `errno`
  - *La variable* `errno` *y los códigos de error estan definidos en* `errno.h`

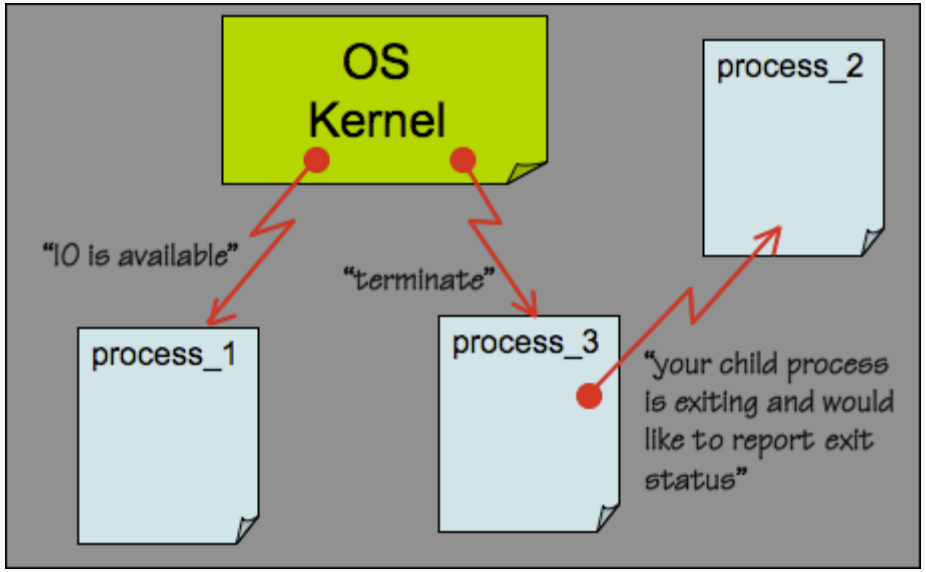
Introducción a la Arquitectura de Linux

- Tipos de error
  - *Fatales*
    - No hay recuperación de estos errores.
  - *No-fatales*
    - Usualmente por falta de recursos. Posible recuperación es intentar más tarde.
- Dos funciones básicas para manejo de errores
  - `char *strerror(int errnum)`
  - `void perror(const char *msg)`

Introducción a la Arquitectura de Linux

Señales

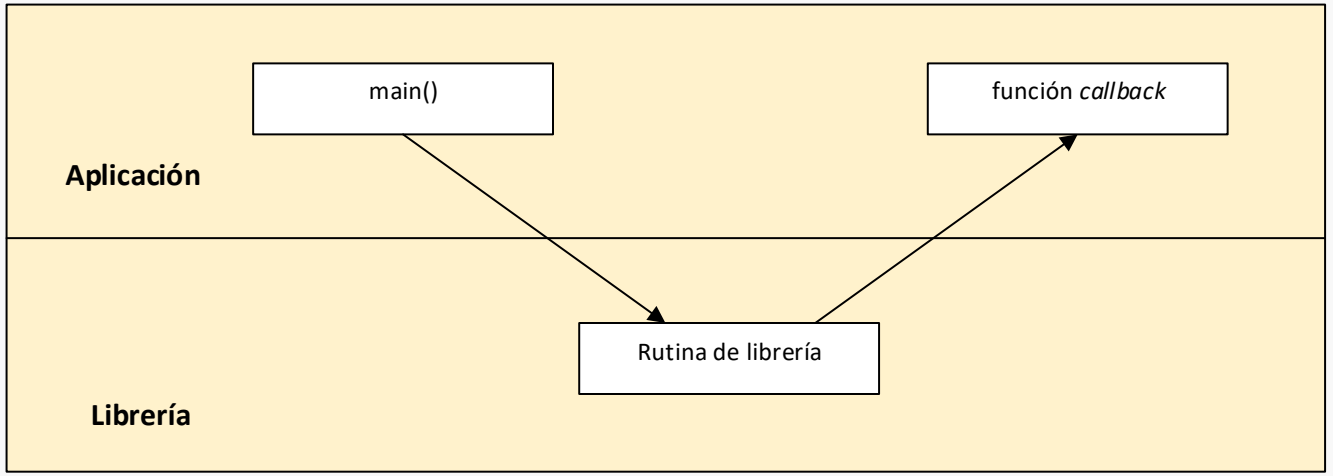
- Mecanismo para notificar a un proceso que algo ha ocurrido
- El manejo de señales en Linux/UNIX hace uso de funciones *callback*



Introducción a la Arquitectura de Linux

¿Qué es una función *callback?*

- Es una función que pasada como argumento a otra función, que se espera que la llame.



Introducción a la Arquitectura de Linux

Tiempo en Linux

- Dos tipos de tiempo:
  - **Tiempo calendario:** valor que contiene el número de segundos desde 00:00:00 Enero , 1970.
  - **Tiempo de proceso (tiempo de CPU):** mide tiempo de ejecución de un proceso
    - Medido el clock ticks
    - 50, 60 o 100 ticks por segundos

1.2 LLAMADAS AL SISTEMA

Llamadas al Sistema y Librerías

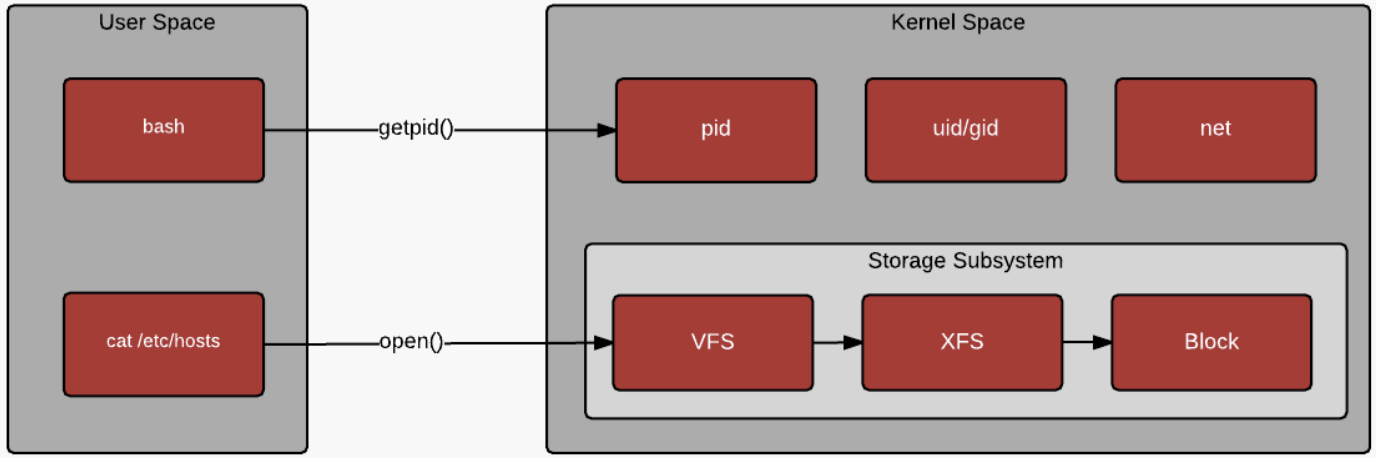
- **Llamada al sistema:** interfaz de programación para acceder a los servicios ofrecidos por el *kernel*.
  - El numero de llamadas al sistema disponibles varia por SO.
- Definidas en lenguaje C.
  - En Linux, la técnica es tener una **función** en la librería standard C con el mismo nombre de la llamada al sistema.
- Para nosotros, las llamadas al sistema son básicamente funciones en C.

Llamadas al Sistema

- SO mantiene un **contexto** para cada proceso corriendo.
- Llamadas al sistema son caras en términos de tiempo, ya que requieren dos cambios de contexto.
  - *Uno para ir de espacio de usuario al kernel*
  - *Otro para ir del kernel a espacio de usuario.*
- **Cambio de contexto:** Es el cambio entre modo KERNEL y de USUARIO.

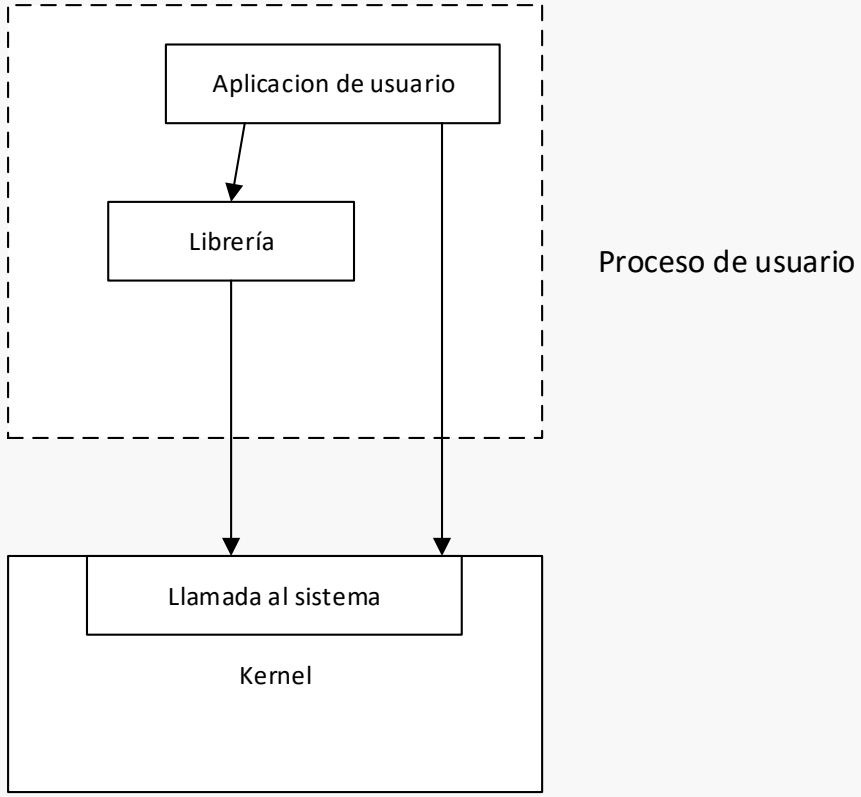
Llamadas al sistema

- CPU tiene dos modos de ejecución:
  - Modo privilegiado → cuando estamos en espacio de kernel.
  - Modo de usuario → cuando estamos en espacio de usuario.



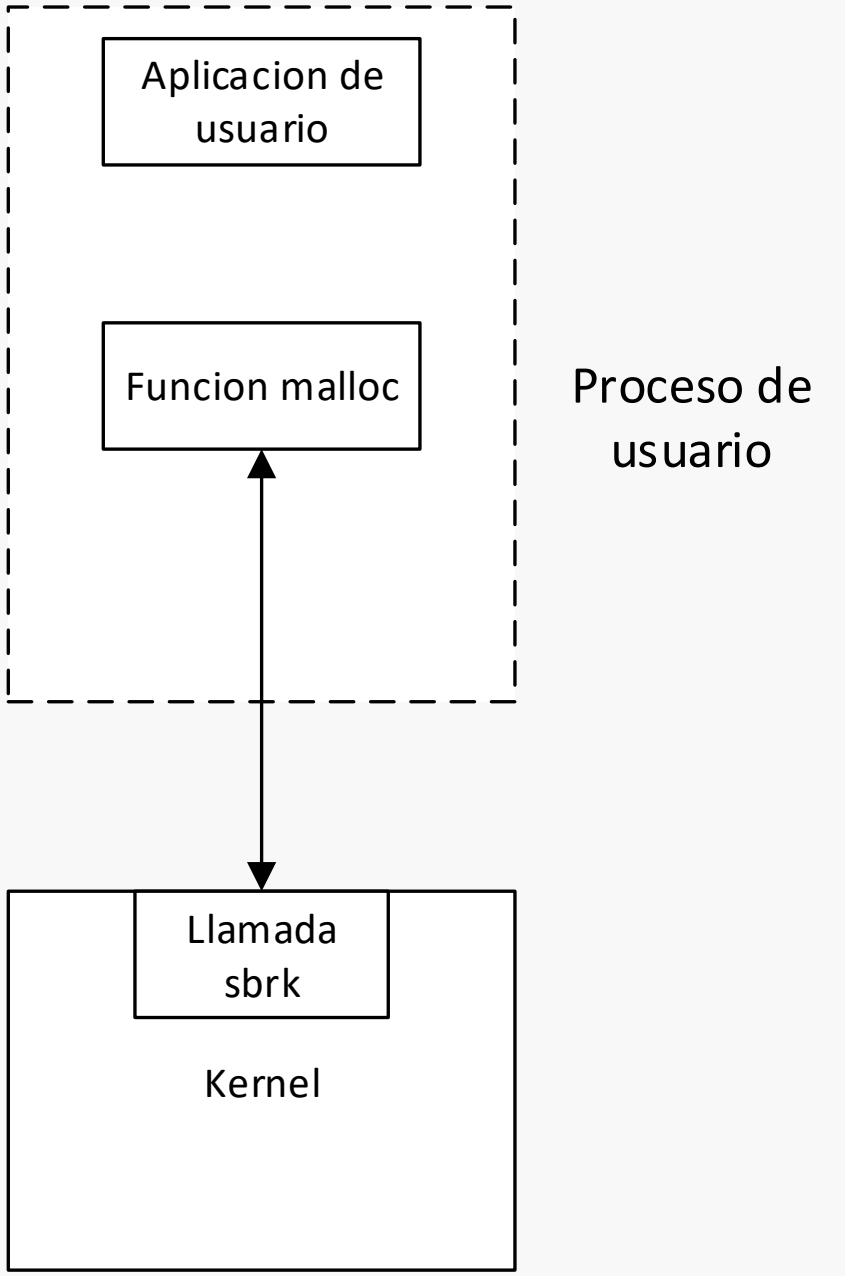
Llamadas al Sistema

- Aplicaciones usar a las llamadas al sistema:
  - *Directamente.*
  - *A través de librerias:*



Llamadas al sistema

- Ejemplo: `write()`
- Aplicación puede llamar a librería o llamada al sistema.
- Rutinas de librería pueden hacer llamadas al sistema.
- Llamadas al sistema suelen tener interfaz mínima, mientras que rutinas de librerías proveen funcionalidad más elaborada.


Aplicacion de usuario
Funcion malloc
Proceso de usuario
Llamada sbrk
Kernel