

# Sistemas operativos

Grupo OS (840) y Gpo 1 (1554)

6to semestre. M.C. Laura Sandoval Montaño.

## Evaluación.

- Trabajos (Ejercicios, cuestionarios, Indagación, etc.) 30 %
- Programas (Escritos en C, ANSI C bajo Linux) 35 %.
- Software 35 %.
- Exámenes parciales

Usuario: fredocgl12 ; Password: N° de cuenta

## Escala

[3, 5.00]	→ 5	3 <→ NP
[6, 6.49]	→ 6	NO APLICA
[6.5, 7.49]	→ 7	EXÁMENES
[7.5, 8.49]	→ 8	FINALES
[8.5, 9.49]	→ 9	
[9.5, ∞)	→ 10	

## getchar()

Antecedentes asignaturas/temas.

- Estructura y programación de computadoras.
- Estructura de datos y algoritmos I y II.
- Linux, - Lenguaje C.

# 1: Introducción a los sistemas operativos

Software → Instructivos  
y de arranque

Programas Datos



Cuantos  
procesos puede  
ejecutar libremente.

## TEMA I - Introducción a los sistemas operativos.

### 1.1 - Concepto y propósito de los sistemas operativos.

Software operativo: Software que permite actuar con un sistema de cómputo o de comunicación.  
(teléfono, Celular),

Sistema operativo: Administra todos los dispositivos y tareas que están conectados entre sí.

Internet de las cosas: Los usuarios del internet son las cosas.

Definición formal de sistema operativo.

Un sistema operativo es software de base que administra los recursos de un sistema de cómputo - comunicación con el objetivo de proporcionar las funcionalidades que dicho sistema provee al usuario.

Generalmente cuenta con una interfaz hacia el usuario y aplicaciones.

Software de base: Cercaño a los dispositivos físicos.  
compiladores, sistema operativo.

## Lógica alambrada

firmware: Programa que es físicamente un chip. (<sup>impreso en</sup> el dispositivo)

BIOS: firmware que se encarga de revisar que dispositivos están conectados.

\* Manejador de base de datos Es SO  $\rightarrow$  Falso

Recursos: Aquellos elementos o aplicaciones para que el SO pueda trabajar.

### • Recursos

Type hardware

Type software

### Recursos

#### Cómputo

##### Hardware

- Memoria.

- Teclado

- Dispositivos de entrada y salida.

- Procesador

- Disco

##### Software

- Programas del usuario.

- Bibliotecas.

- Procesos.

#### Comunicaciones

##### Hardware

- Modem.

- Altavoz/Bocina

- Drivers de hardware

##### Software

- Mensajes.

- Llamadas

- Drivers de software.

Los más importantes.

Proceso: Un programa en ejecución.

Ex:

```
#include <stdio.h>
int main()
{
    form();
    print("Hola amigos");
}
```

driver: Programas de bajo nivel que generalmente se encargan de manipular el hardware, son invocados por el procesador cuando son muy bajo nivel y con el sistema operativo.

Funcionalidades de un sistema de computo-comunicación

- Permite al usuario almacenar, recuperar y procesar datos.
- Administra las tareas del procesador para que ejecute programas del usuario y aplicaciones de manera eficaz.
- Utilizar el hardware del sistema de forma eficiente.
- Proporcionar una interfaz de acceso a dispositivos y datos remotos, conectados a través de líneas de comunicación.
- Generalmente cuenta una interfaz hacia el usuario y aplicaciones. Puede ser a través de una interfaz de comando. También por interfaz gráfica.

Unix : cd ..\alfredo\sistemasoperativos

Ms Dos : cd ..\alfredo\sistemasoperativos

- Las primeras computadoras no utilizaban sistemas operativos.
- Los primeros sistemas operativos solo podían hacer un proceso a la vez.

### Programas de Usuario y Aplicaciones

Compiladores | Interpretes de comandos

Software aplicativo

### Sistema Operativo

Software de base

### Lenguaje de máquina

Microprogramación

Hardware

Dispositivos físicos

→ Instrucciones en lenguaje máquina que le dice al procesador que hacer, incrusta en algo físico (no se puede borrar) un ejemplo es el bios.

## 0.5: Operate System

### 1.2 Evolución de los Sistemas Operativos.

- 1940: No existían sistemas operativos, entonces directamente el usuario a través en lenguaje máquina le indicaba a la computadora lo que debía hacer, en el lenguaje binario, generalmente para calcularlos, no para almacenar archivos.
- 1950:
  - Surge el primer sistema operativo para una computadora IBM 704 (en el año 1956). Ya no se depende que el usuario sea un especialista para utilizar el hardware.
  - Los computadoras eran electromecánicas, usaban bulbos.

1950:

#### Hardware y software

- Computadoras electromecánicas (bulbos)
- S.O. IBM 704

#### Manejo de procesos

- Una tarea a la vez (unitarea).
- Procesamiento por lotes (batch).

1960

- Ya no son computadoras electromecánicas sino electrónicas porque utilizan transistores
- S.O. IBM 360 (OS/360)
- S.O. Multics
  - ↓
  - Por multitareas.
- En 1969 surge UNIX

- Procesamiento por lotes.
- Multiprogramación O multitareas.
- Multiusuarios.
- Usuarios interactivos.
- Memoria virtual.
- Sistemas de tiempo real.

Usario interactivo: Interacción en línea.

Interacción en tiempo real: Respuesta inmediata.

1970

## Hardware y Software

## Manejo de procesos

1970

- Auge de los circuitos integrados reduce el espacio de las computadoras y surgen las minicomputadoras.
- Se inicia la creación de las microcomputadoras así como los microprocesadores (Todo se reduce en tamaño pero no en capacidad).
- Surge memoria de semiconductores.
- Almacenamiento masivo
- Redes de área local
- Se reescribe UNIX en lenguaje C.

• Multiples modos de operación en un solo sistema.  
(Multiusuario y multiprogramación)

↓  
Varios usuarios atendidos en el sistema

1: Verdadero.

2: Falso.

3: Verdadero

4: Verdadero

5: Verdadero

Linux funciona en cualquier procesador.

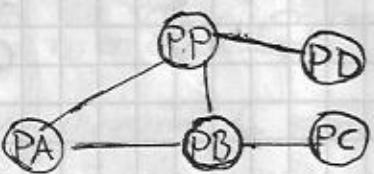
	Hardware y software	Manejo de procesos
1980	<ul style="list-style-type: none"><li>• El auge de las computadoras personales (PC) (Personal Computer).</li><li>• Surgen estaciones de trabajo y servidores</li><li>• microprocesadores (Gran avance)</li><li>• Redes de computadoras metropolitanas.</li><li>• Surge el internet.</li><li>• Ms-Dos</li><li>• Windows.</li><li>• MacOs</li></ul>	<ul style="list-style-type: none"><li>- Ademas de los anteriores.</li><li>- Surge el Cómputo distribuido.</li><li>- Surge el modelo Cliente-Servidor.</li></ul>
1990	<ul style="list-style-type: none"><li>• Surge la arquitectura masivamente paralela.</li><li>• Utilizaban las supercomputadoras.</li><li>• La www crece rápidamente.</li><li>• Surge el sistema operativo Linux (en 1991).</li></ul>	<p><u>Todos</u></p> <ul style="list-style-type: none"><li>- Se difunde más la Computación distribuida.</li><li>- Surge los protocolos de sistemas abiertos.</li></ul>
2000	<ul style="list-style-type: none"><li>• Dispositivos inteligentes.</li><li>• La nube (AWS)</li></ul>	<p><u>Web</u></p>
2010	<ul style="list-style-type: none"><li>• Internet de las cosas.</li><li>• Auge de la inteligencia artificial (forma más global) para aplicaciones.</li><li>• Open Source (Github y Red Hat)</li></ul>	
2020	<ul style="list-style-type: none"><li>• Redes 5G (Incremento IoT)</li></ul>	

## 1.3: Estructura de los sistemas operativos

### \*Estructura Monolítica

Un solo programa realiza las funciones básicas del sistema operativo.

- Está formado por funciones y procedimientos.

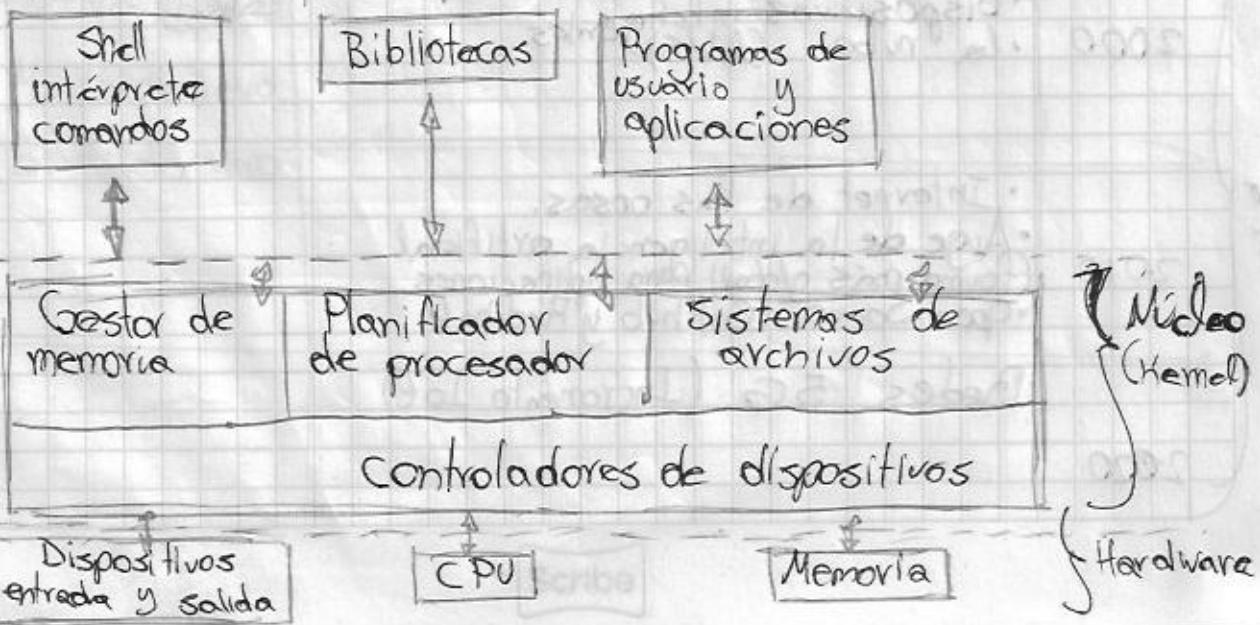


Es un caos organizado, por ser un solo programa. Todo el código está en memoria principal.

Ejemplo:

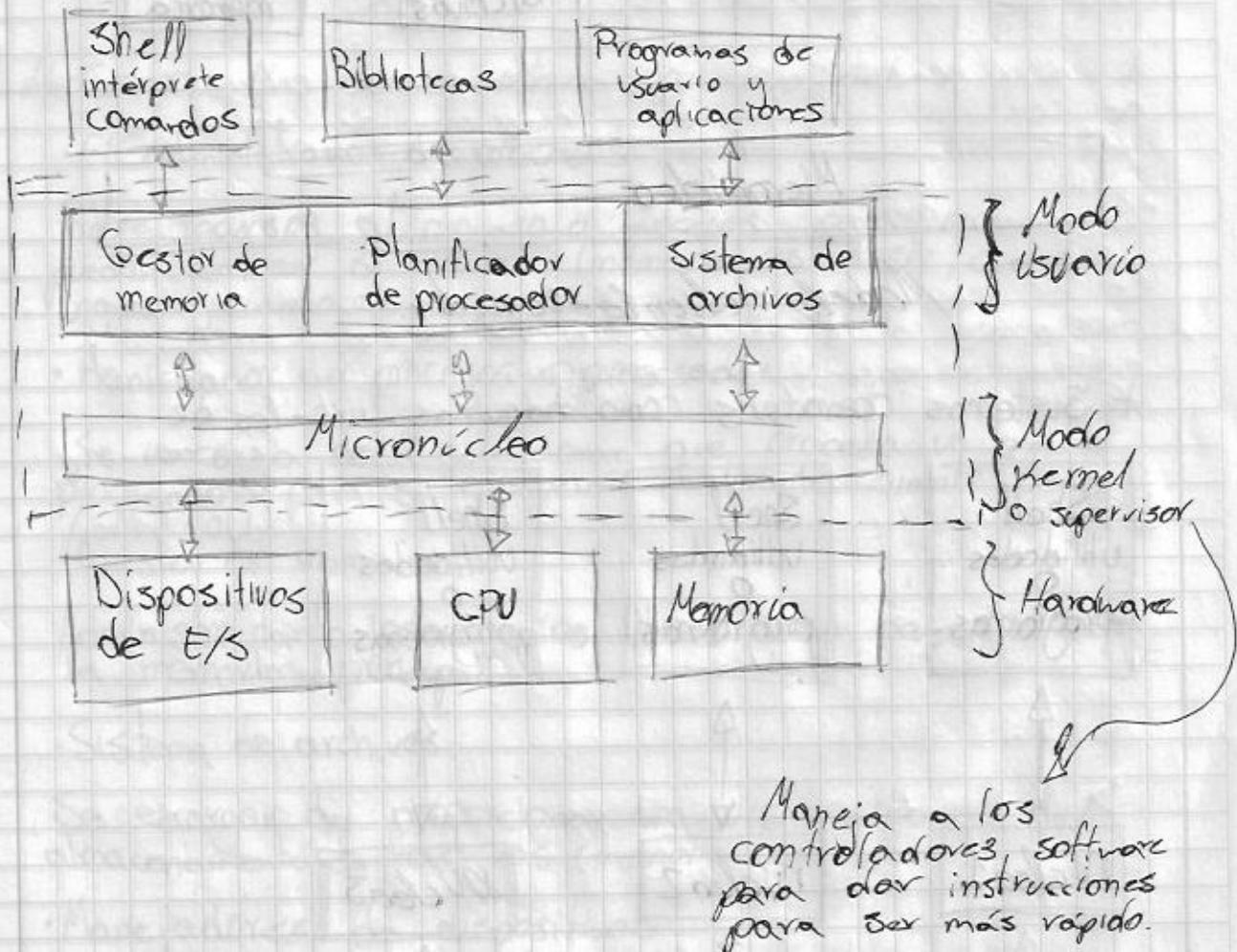
- MS-DOS (Sistema operativo en disco).
- Ms DOS
  - ↳ System
  - ↳ operating
  - ↳ Microsoft Disk
- Linux | Unix

### \* Estructura por capas



Terminales: C-Shell  
K-Shell  
B-Shell

## \* Estructura de Micrónucleo.



// En Linux se pueden aumentar o disminuir el número de procesos del sistema operativo.

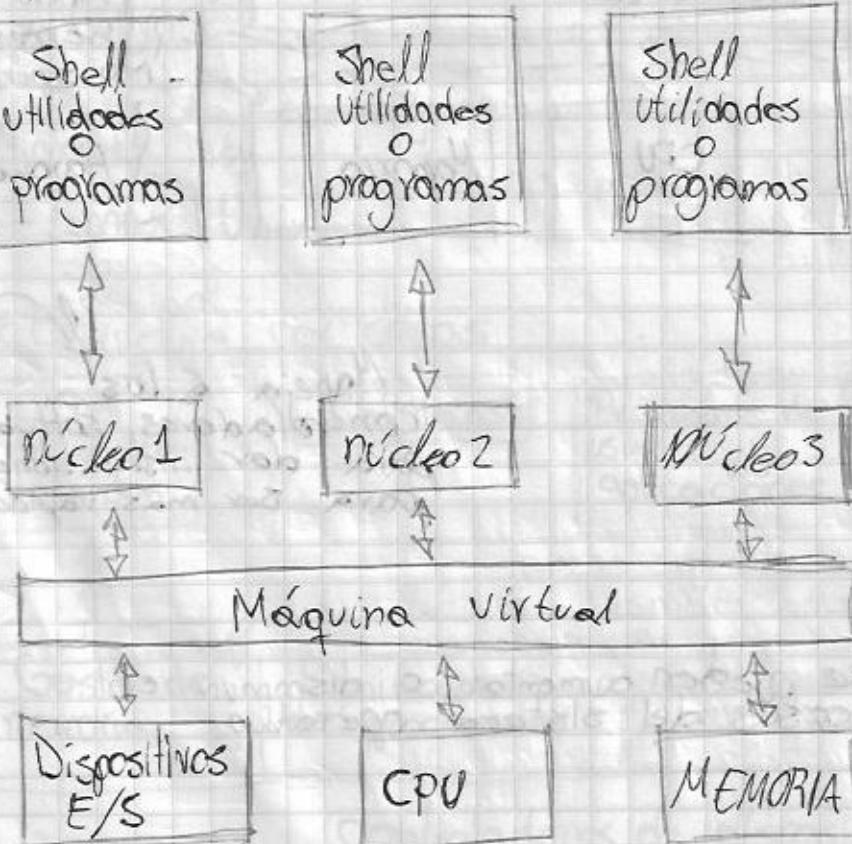
\* Variante estructura micronúcleo.

✓ Solo se cargan por indicaciones del micronúcleo.



: Modelo cliente-servidor.

\* Sistemas operativos como máquinas virtuales de tipo hardware (No VirtualBox)



Eg. en si  
el Kernel  
de un  
Sistema  
Operativo.

administrador = Gestor

Alfredo

## \* Maquina virtual de software:

Para abstraer o virtualizar un ~~software y hardware~~ sistema total tanto software como hardware, cuyo objetivo es integrar varios sistemas operativos dando la sensacion de ser varias maquinas diferentes.

## 1.4 Componentes de un sistema operativo actual.

### • El administrador de procesos

Provee recursos al proceso o procesos para que puedan realizar su tarea. (memoria, estados, cuando ingresan, cuando se suspenden)

### • Planificador de procesos y procesador

Se encarga de seleccionar que proceso va con el procesador para su ejecucion y durante cuento tiempo.

### • Gestor de memoria.

Administra tanto los espacios libres como ocupados de la memoria principal.

### • Sistema de archivos.

Se encargan de proporcionar una vista uniforme del almacenamiento de archivos (memoria secundaria).

### • Manejadores de dispositivos

Oculta las caracteristicas especificas de cada dispositivo y ofrece servicios comunes a todos.

// procesos se en memoria principal.

• Las computadoras personales las definió IBM.

obs. 2/A

- Llamadas al sistema.

Son rutinas que ordenan al sistema operativo que desempeñe un trabajo referente a manipulaciones detalladas del hardware y software.

- Sistemas de comunicaciones.

A través de estos sistemas de comunicaciones se encargan de controlar el envío y recepción de datos a través de las interfaces de red.

- Sistemas de protección.

Es el mecanismo que controla el acceso de los programas o los usuarios a los recursos del sistema o especificar los controles de seguridad a realizar.

## 1.3. Consideraciones de diseño de un Sistema operativo.

En el diseño se ve como va a ser la convivencia de los procesos que va a estar administrando el SO.

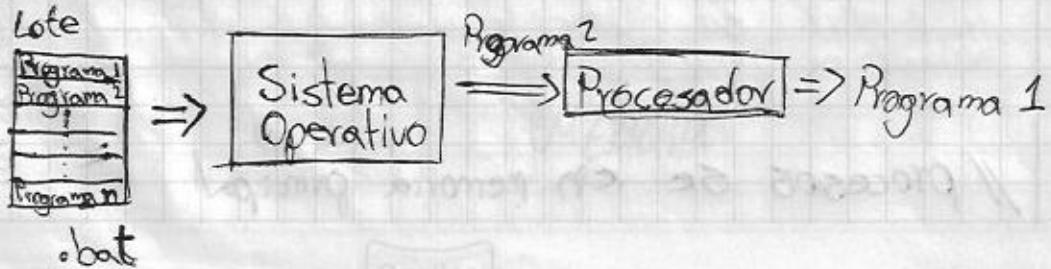
El más simple.

> Sistema operativo de tarea única y único usuario.

En la memoria hay un solo SO y un solo proceso.

Ejemplos: CP/M y primeras versiones de MS DOS

> Sistemas operativos por procesamiento por lotes.  
o batch.



· multiprocesamiento ≠ multitarea

\* El que ejecuta los proceso es el procesador

\* El SO decide que proceso se le asigna al procesador.

> Sistema operativo multitarea o multiprogramación.

Se manejan varias tareas o procesos de manera simultánea o concurrente.

El sistema maneja la memoria tiene los procesos, los procesos compiten por los recursos del sistema, incluso la memoria, pero el más importante es el tiempo del procesador.

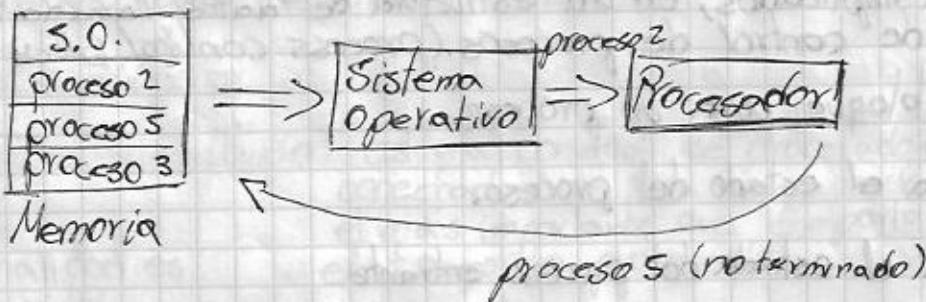
- con un solo procesador solo 1 proceso puede estar en ejecución -

- con dos o más unidades de procesamiento puede haber dos o más procesos ejecutandos (1 en cada unidad de procesamiento)

↳ Esto es Multiprocesamiento.

En todo sistema de multitarea ocurre lo que es el cambio de contexto.

Cambio de contexto



\* " " debe ser multitarea → Verdadero.

\* En un sistema de multiprocesamiento debe ser multiusuario → Falso

· Unidad de procesamiento: Pueden los núcleos del procesador.

• Registros.

• Program Counter: Apuntador de programa, continua después de donde se quedó.

Cambio de contexto: Continuar un proceso donde se quedó.

Continuar un proceso que f no fue terminado,

Continuar un proceso que no fue terminado, justamente donde se quedó.

El cambio de contexto

Preservar el estado del proceso saliente y reestablecer el estado del proceso entrante.

Cuando se realiza  
Cambio de contexto.

• Cuando se

1- Del proceso saliente se guardan todos sus valores asociados, como los de los registros y direcciones de memoria implicados, en una estructura de datos llamada: bloque de control de procesos. (process control block (PCB)).  
Un bloque para un proceso.

2- Cambia el estado del proceso.

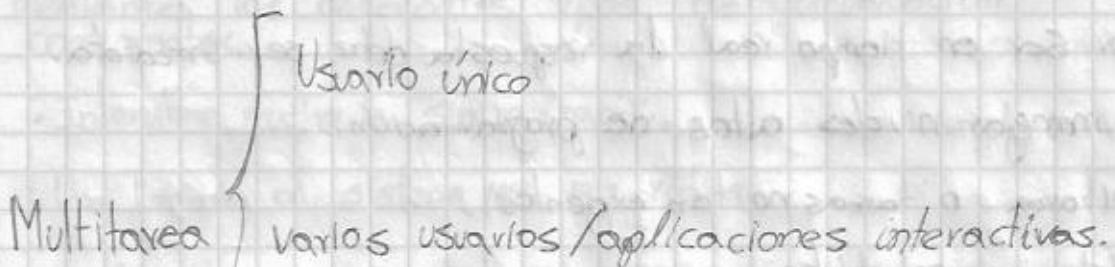
3- Cambia el estado del proceso entrante.

4- Se cargan los valores de los registros y direcciones de memoria asociados al proceso entrante, localizados en su estructura PCB.

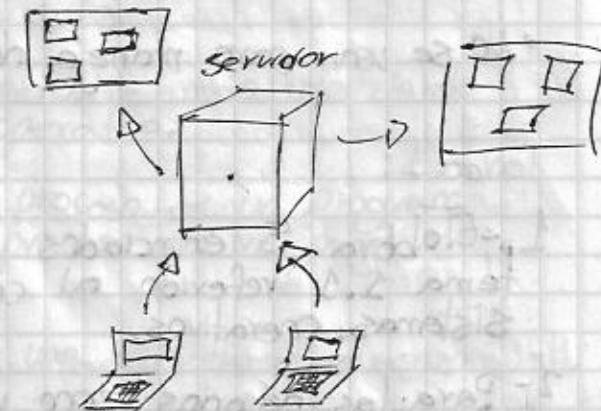
Se cargan los valores de los registros y direcciones de memoria asociados a los procesos en ls. estructura PCB

- Un usuario único puede mandar a ejecutar varias tareas.
- SO: Software que administra los recursos para que el sistema de cómputo sea funcional.

Procesos interactivos: Respuesta inmediata del usuario.



Userivo único  
Puede darse con



> Sistemas operativos distribuidos.

Trabajan sobre sistemas distribuidos.

Un sistema operativo

Sistema distribuido: Es una conexión de procesadores conectados en red. Comparte recursos, el más importante que comparte es el tiempo de procesador.  
 ↓  
 Su finalidad es compartir recursos.

> Desempeña las mismas funciones que un sistema distribuidos normal pero sus recursos van a estar distribuidos en una red.

# 1.- Introducción a los sistemas operativos

## > Sistemas operativos en tiempo real

- Por ser en tiempo real la respuesta debe ser inmediata.
- No manejan niveles altos de programación.  
(Unitareas o tareas no tan exigentes).
- Sistemas embebidos.
- No se usan para manejo de mucha información en archivos.

## Tarea 1:

1.- Elabora 5 enunciados de verdadero y falso del tema 1.1 referido al concepto y propósito de los sistemas operativos.

2.- Para las décadas 2000 y 2010 <sup>a la fecha</sup> incluye una tecnología de la época y un tipo de manejoamiento de procesos.

3.- De la estructura de los sistemas operativos, indagar cual es la estructura del Kernel en Unix/Linux.  
(Decir como se comunican los componentes).

4.- Elaborar un cuadro sinóptico de los diferentes diseños de un sistema operativo (vistas en 1.5).

## Tema 2. Administración de procesos

El objetivo es que el alumno clasifique los procesos mediante los diferentes tipos de comunicación concurrentes.

- Interviene mucho la sincronización.
- Una llamada al sistema es una rutina.

### 2.1: Procesos. Concepto y estructura.

El proceso es el recurso software más importante que administra el sistema operativo.

- Generalmente se dice que un proceso es un programa en ejecución pero no todo proceso es un programa en ejecución.
- Otra definición: Un proceso es una actividad asíncrona.
- " " : Es el espíritu animado de un procedimiento.
- " " : Es el centro de control de un procedimiento en ejecución.
- " " : Es la entidad a la que se le asignan los procesadores.

¿Qué es un programa y qué es un proceso?

↓  
• Es estático porque ~~estático~~ es un archivo, está en el almacenamiento secundario.

• Cuando se ejecuta se vuelve dinámico porque pasa a memoria principal.

↓  
• Es dinámico, reside en memoria principal.

## Tema 2: Administración de procesos

### Creación de procesos.

• De manera general hay cuatro formas.

1.- Como parte del arranque del sistema operativo.

// En linux el primer proceso que se crea es init()

2.- Dando la orden de ejecución de un programa o aplicación.  
(por línea de comando o interfaz gráfica).

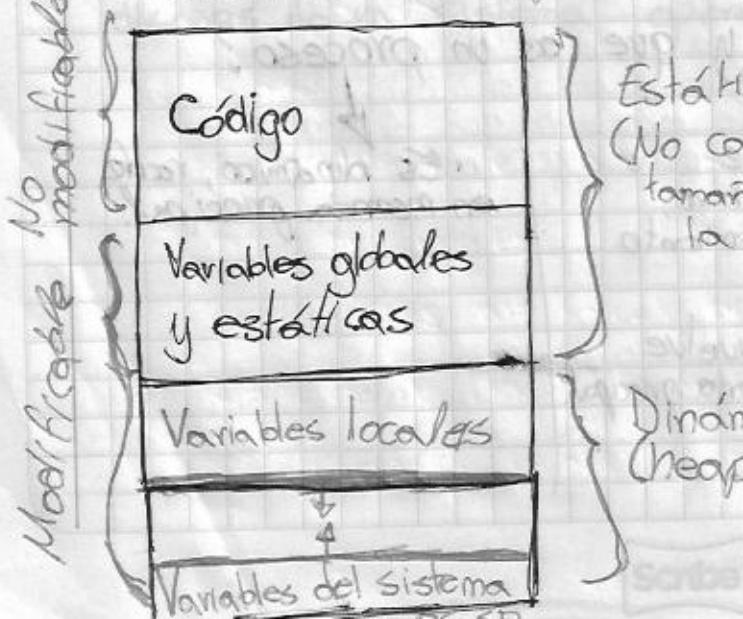
3.- A través de un programa en ejecución utilizando llamadas al sistema.

// Para crear un proceso se utiliza la llamada al sistema fork() de linux/unix \*//

4.- Como parte del procedimiento por lotes en un sistema que lo realice de manera automática.

Cuando se crea un proceso se le asignan recursos del sistema, el tiempo de procesador (más importante), memoria principal, dispositivos de entrada y salida, entre otros.

Estructura de un proceso. //Se divide en tres partes \*//



Estáticas  
(No cambian su tamaño durante la ejecución del proceso)

Dinámico  
(heap)

# BLOCK CONTROL PROCESS

## Tema 2: Administración de procesos.

Bloque de control de procesos. (BCP o PCB).

// \* Se utiliza en el cambio de contexto \*

\* Es un bloque por cada proceso \*

\* La información de los registros, de las variables del sistema están en el PCB \*/

Generalmente el BCP contiene:

1- El estado actual del proceso.

2- Un identificador único (PID) /\* de proceso \*/  
a init() se le asigna el PID 1.

3- Un apuntador hacia el proceso padre. (El que lo creo).

Todo proceso es creado por un proceso excepto init(). Todos

4- Apunadores a los procesos hijo

5- Prioridad del proceso. Generalmente un S0 tiene prioridad

pri

6- Apunadores hacia zonas de memoria del proceso.

7- Apunadores a los recursos asignados al proceso

8- Una área de salvaguardia de los registros.

---

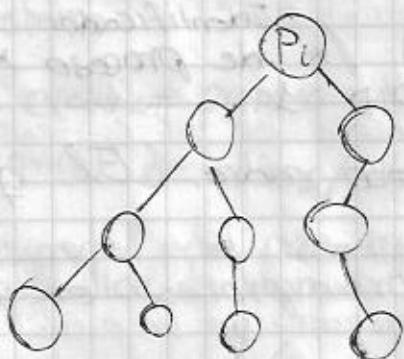
\* Variables del sistema: Variables del sistema operativo para administrar el proceso. Ejemplo: program counter, stack counter

## Tema 2-Administración de procesos.

q) El procesador en el que se está ejecutando.

La BCP es la entidad que define un proceso al Sistema operativo. Es muy importante su manejo y de hecho muchos sistemas de cómputo tienen un registro de hardware que siempre apunta hacia el PCB del proceso que se está ejecutando. Esto para que el acceso sea inmediato.

Estructura jerárquica de procesos.



Procesos en linux.

Los procesos en linux tienen la estructura antes vista (la que está dividida en tres). Debe ser el PCB se llama tabla de procesos, donde específicamente linux maneja los siguientes campos:

- 1- Estado
- 2- Apunadores a memoria ocupada.
- 3- Campo de señales.
- 4- Temporizadores para cálculo de prioridad dinámica.
- 5- Parámetros de planificación.
- 6- Identificador de usuario. (UID)

## Tema 2: Administración de procesos.

7: Identificador de proceso (PID).

8: Descriptores de eventos.

Todos los proceso excepto init(), son creados por la llamada al sistema fork().

```
#include <sys/types.h>
```

```
#include <unistd.h> // Para que reconozca fork()
```

```
pid_t fork(); // No tiene argumentos y nos regresa un identificador de tareas.
```

```
#include <sys/types.h>
```

```
#include <unistd.h> // Unix standar
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    fork(); /* Se crea proceso hijo */  
    printf("hola amigos\n");
```

```
    /* El printf lo escribe el padre  
    return 0; el hijo */
```

```
}
```

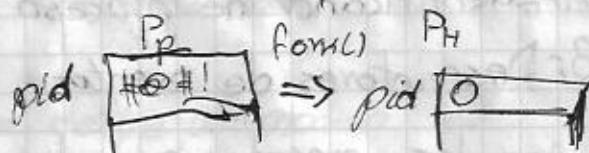
- Solo lo que está abajo del fork();
- No se sabe cuál se ejecutará primero.

Tanto el proceso padre como el proceso hijo ejecutan el mismo código ubicado en la memoria original (No se duplica el código)

\* Además del código comparten los archivos, hola amigo saldrá dos veces en el manito.

## Tema 2: Administración de procesos

Tanto variables  
globales y locales  
se copian.



```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
```

```
int main()
```

```
{
```

```
    int pid;
```

```
    pid = fork();
```

```
/* En este punto, fork(); ya creó el proceso hijo. */
```

```
    if (pid == 0)
```

```
{
```

```
    printf("Soy el padre.");
```

```
}
```

```
else
```

```
{
```

```
    printf("Soy el hijo.");
```

```
}
```

```
return 0;
```

```
} //main
```

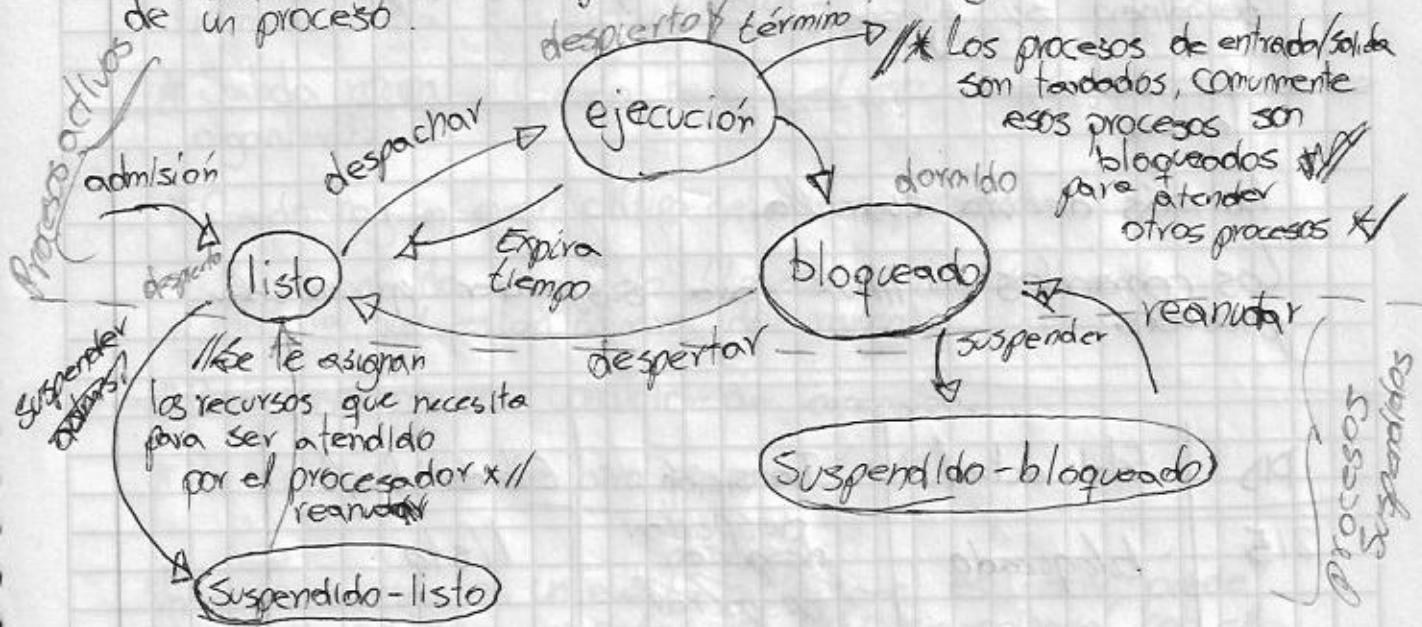
\$ kill -9 : Para matar a un proceso de a de veras.

7

## 2. Administración de procesos

### Estados de un proceso.

De manera general se pueden decir que hay tres estados de un proceso.



• Un proceso suspendido no puede proseguir sino hasta que lo reanuda otro proceso

• Razones para suspender a un proceso:

\* Se vea que el sistema está funcionando mal y pueda fallar.

\* Un usuario que está al pendiente de los resultados de su proceso, lo puede suspender para ver sus resultados parciales y posteriormente puede reanudarlo.

\$ kill -9 1037

\$ ps -x : Ver relación de procesos (Cuantos procesos se han lanzado)

Todos los que han existido desde que encendiste la PC.

\* Un proceso activo puede estar activo : Verdadero

## 2.- Administración de procesos

- \* Si se ve que hay mucha carga de trabajo y se requiere que un proceso importante termine, se requiere suspender el proceso para que los importantes terminen su ejecución.

Tareas:

No más de una cuartilla.

Los comandos en Linux para suspender y reanudar procesos.

PID	Estado actual	Transición	Estado final
215	bloqueado	despertar despertar despertar	listo
217	listo	Suspender	suspendido-listo
215	listo	despachar	ejecución
218	suspendido- bloqueado	reanudar	bloqueado
215	ejecución	bloquear	bloqueado
220	Ejecución	expira tiempo	listo

## 2. Administración de procesos

- Razones para que los procesos terminen.
  - \* Cuando un proceso termina, generalmente pasado un tiempo libera sus recursos.
  - \* Salida normal (Cuando termina el proceso por su algoritmo).
  - \* Cuando hay un error crítico del mismo proceso.
    - // Violación de segmento (Acceder a localidades de memoria que están fuera del rango).
  - \* Poder ser por una condición de excepción.
  - \* Por recibir señal de otro proceso (kill).

Una interrupción es un evento asincrónico, es decir, puede ocurrir en cualquier momento; no está sincronizado con el reloj del sistema ni con el ciclo de ejecución de instrucciones del procesador.

Cuando el SO detecta una interrupción que puede manipular, realiza lo siguiente:

1. \* El sistema operativo toma el control ante la recepción de una señal de interrupción.
2. - El SO guarda el estado del proceso interrumpido.
3. // El estado se aloja en el PCB
3. - El SO analiza la interrupción y transfiere el control a la rutina apropiada para atenderla.
4. La rutina del manejador de interrupciones procesa la interrupción.
5. Se restaura el estado del proceso interrumpido.  
(Del que estaba en ejecución)

Cuando hay una llamada al sistema es una interrupción

2-Administración de procesos

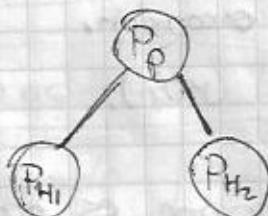
getpid

Pn: Proceso  
nicho

b- Continua la ejecución del proceso interrumpido.

Llamadas al sistema de manejo de procesos.

- fork():
- Crea un proceso con el mismo código del proceso que lo creó. (PID del hijo)
  - Regresa un valor al proceso padre y otro al proceso hijo (P)
  - Si no puede crear un proceso regresa -1



Este caso:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int pidH1, pidH2;  
    pidH1 = fork();
```

```
    if (pidH1)
```

```
        pidH2 = fork();  
        if (pidH2)
```

```
{
```

```
    printf("Soy el padre y ya cree dos hijos")
```

```
    printf("Sus PID son: %d y %d", pidH1, pidH2);
```

```
}
```

else

{

else

{

```
    printf("Soy el hijo 1 y mi  
    pid es: %d, getpid());
```

```
    return 0;
```

} // fin del main

O: falso

otro número: verdadero

El hijo no  
sabe cuál es  
su pid

## - Administración de procesos

### Características del proceso padre versus proceso hijo.

Al momento de crear un proceso con `fork()`, el nuevo proceso se le asigna un identificador (PID), el cual debe de ser único.

#### - Qué comparten:

- \* Código. (la misma localidad de memoria).
- \* Archivos abiertos por el padre antes de crear al hijo.

#### - Qué no comparten:

- \* Variables globales.
- \* Variables totales.
- \* Variables del sistema.

El hijo ocupa otra área de memoria para estas variables.

• Si el padre termina, los hijos son adoptados por otro proceso de una jerarquía más grande.

Ej:

## 2. Administración de procesos

### Ejercicio

Elabora un programa en C cuya jerarquía de procesos sea de un padre y un hijo. El proceso padre definirá un arreglo de cinco enteros, el hijo escribirá el menor del arreglo.

#

#

int main()

{ int arr[5] = {25, 17, 8, 40, 13};

int pidH;

pidH = fork();

if (pidH)

{ printf("Soy el padre \n");

arr[3] = 1; // Solo se modifica el del padre.

}

else

{

int i, min = arr[0];

for (i = 1; i &lt; 5; i++)

{ if (min &gt; arr[i])

min = arr[i];

} // fin for

printf("El menor es: %d \n", min);

} // fin else

return 0;

{ // fin main

P<sub>p</sub>  
arr[0] = 25  
arr[1] = 17  
arr[2] = 8  
arr[3] = 40  
arr[4] = 13  
pidH = 3061

P<sub>H</sub>  
arr[0] = 25  
arr[1] = 17  
arr[2] = 8  
arr[3] = 40  
arr[4] = 13  
pidH = 0  
i = #0! ✓ 2345  
min = 25, ✓ 8

## 2.- Administración de procesos

Llamada wait.

wait(): A diferencia de fork() y getpid(), este si necesita ~~regresa~~ argumentos, necesita la biblioteca

```
#include <sys/wait.h>
```

```
pid_t wait(int *stat_loc);
```

Espera a que termine a su hijo para terminar, si tiene varios hijos espera a que termine el primer hijo.

```
wait(NULL); wait(NULL);
```

Llamada exit.

exit(): Requiere la siguiente biblioteca:

```
#include <stdlib.h>
```

```
void exit(int status);
```

exit tiene dos funciones, cuando un proceso maneja sentido del hijo al padre, sólo una contando de 0 a 255

es el equivalente del return de una función, pero exit es para un proceso

Llamada pipe()

Crea un canal de comunicación entre proceso padre y proceso hijo.

Se puede decidir que uno escriba y otro lea en cualquier orden



>> CC : Comimientos de bits

## 2: Administración de procesos

// Para que el padre lo imprima

#  
#  
#

int main()

{ int arr[5] = {

int pidH, valor;

pidH = fork();

if (pidH)

{

print("Soy el padre");

wait(&valor);

printf("El minimo es : %d", valor >> 8);

}

else

{

int i, min = arr[0]; // sleep(1);

for (i = 1; i < 5; i++)

{

if (min > arr[i])

min = arr[i];

} // fin for

exit(min);

} // fin else

return 0;

} // fin main

exit

010101011000

wait

0101011000000000

int main()

{ int arr[5] = {

int pidH, valor;

pidH = fork();

if (pidH)

{

print("Soy el padre");

wait(&valor);

printf("El minimo es : %d", valor >> 8);

}

else

{

int i, min = arr[0]; // sleep(1);

for (i = 1; i < 5; i++)

{

if (min > arr[i])

min = arr[i];

} // fin for

exit(min);

} // fin else

return 0;

} // fin main

X = 4

X = X << 2;

X será 16

Tarea: Elaborar un programa en C que como estructura de procesos sea el proceso padre y dos procesos hijo, los procesos hijo deben coexistir. Luego luego el padre debe crear los dos hijos. El proceso padre creará un arreglo de 10 elementos enteros, valores van del 0 al 255, el P1 calculará el mínimo, se lo enviará al padre para que lo escriba, P2 calculará el mayor y se lo enviará al padre para que lo escriba. Deben de aparecer 2 wait. //En un solo printf() el menor y el mayor.

## ? Administración de procesos.

### Clases de interrupciones

Existen seis clases de interrupciones.

1: Interrupciones del llamado al sistema o supervisor.

2: Interrupciones de entrada y salida

Cuando se va a usar un dispositivo de entrada y salida

Se generan al tratar de hacer una operación con dispositivos de entrada y salida. Pasa a bloqueado.

3: Interrupciones externas de entrada y salida.

Hay una incidencia, el movimiento del ratón.

4: Interrupciones de verificación de la máquina.

Se ocasiona por el mal funcionamiento del hardware, por ejemplo, cuando falla una sección del disco o que se llegue a dañar un módulo de memoria principal.

5: Interrupciones de verificación del programa.

Ocurre por mucha frecuencia entre los programadores, ejemplo: dividir entre cero, leer mal una cadena, ejecutar un código con algo inválido, localidad de memoria incorrecta.

6: Interrupciones críticas.

Por cuestiones externas

Se cae el café, se cae el equipo etc.