

Sistemas Operativos

Introducción

Marcelo Arroyo - Laura Tardivo

Dpto. de Computación - FCEFQyN - Universidad Nacional de Río Cuarto

2017

Temas a estudiar en el curso

- Conceptos generales sobre sistemas operativos
 - Sistemas de computación (hardware, software)
 - Procesos, comunicación entre procesos, ...
 - Memoria
 - Entrada/salida y Sistemas de Archivos
 - Seguridad: autenticación, cifrado, ...
- Diseño e implementación
 - Taller: extensiones a un SO educativo: **xv6**
- Bibliografía
 - *Operating System Concepts* (6^{ta} edición). Silverschats.
 - *Modern Operating Systems - Design and Implementation*. A. Tannenbaum.
 - *Fundamentos de Sistemas Operativos*. G. Wolf, E. Ruiz, F. Bergero, E. Meza.
 - *Varios manuales técnicos*: ELF, IA32, ...

Sistema de computación

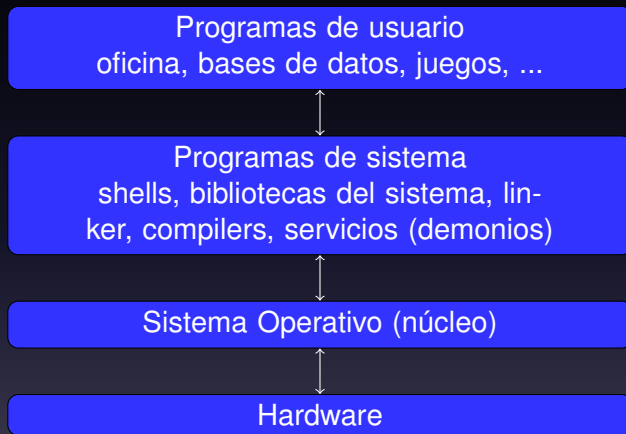


Figura Sistema de computación

Historia y evolución

- 1- Primera generación (1945-1955): Tubos de vacío y tableros de conexión.
- 2- Segunda generación (1955-1965): Transistores, sistemas por lotes (batch).
 - Language control jobs: FMS (Fortran Monitor System, IBM IBSYS).
- 3- Tercera generación (1965-1980): Circuitos integrados, multiprogramación.
 - CTSS (en IBM 7094), MULTICS, UNICS (UNIX)
- 4- Cuarta generación (1980-1990): CI (LSI), tiempo compartido y computadoras personales.
 - IBM PC (MS-DOS, XENIX), Apple Machintosh (MacOS), IBM OS2, ...

Algunos conceptos

- **Sistemas monotareas y por lotes**
 - Ejecuta un proceso (job) a la vez
 - Lenguaje de control de jobs (Load, Run, Pause, Cancel, Finish, ...)

Algunos conceptos

- **Sistemas monotareas y por lotes**

- Ejecuta un proceso (job) a la vez
- Lenguaje de control de jobs (Load, Run, Pause, Cancel, Finish, ...)

- **Multiprogramación**

- Ejecuta varios procesos
- Requiere particionar la memoria y gestionar el acceso a los recursos
- Una operación de E/S suspende el proceso y el SO activa otro (scheduling)
- Aprovecha el paralelismo entre E/S y uso de CPU
- Requiere protección de memoria y al menos dos modos de operación

Algunos conceptos

- **Sistemas monotareas y por lotes**

- Ejecuta un proceso (job) a la vez
- Lenguaje de control de jobs (Load, Run, Pause, Cancel, Finish, ...)

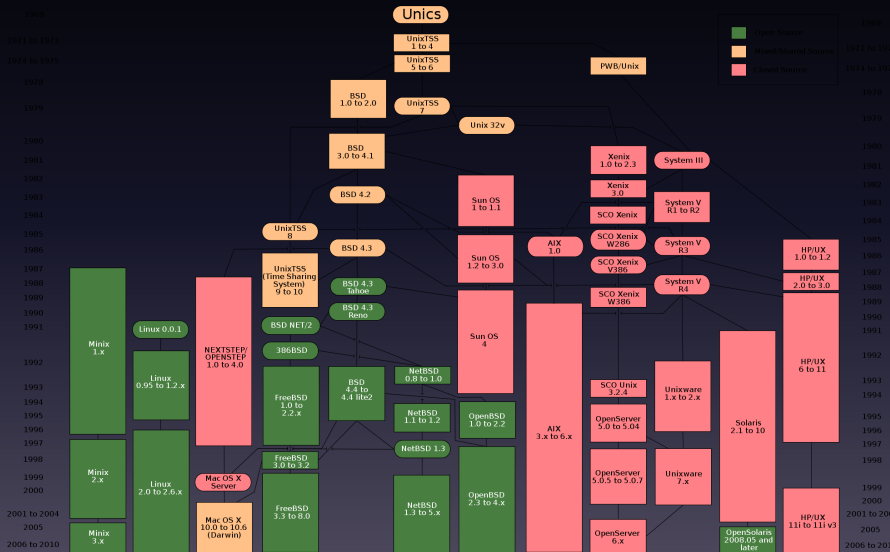
- **Multiprogramación**

- Ejecuta varios procesos
- Requiere particionar la memoria y gestionar el acceso a los recursos
- Una operación de E/S suspende el proceso y el SO activa otro (scheduling)
- Aprovecha el paralelismo entre E/S y uso de CPU
- Requiere protección de memoria y al menos dos modos de operación

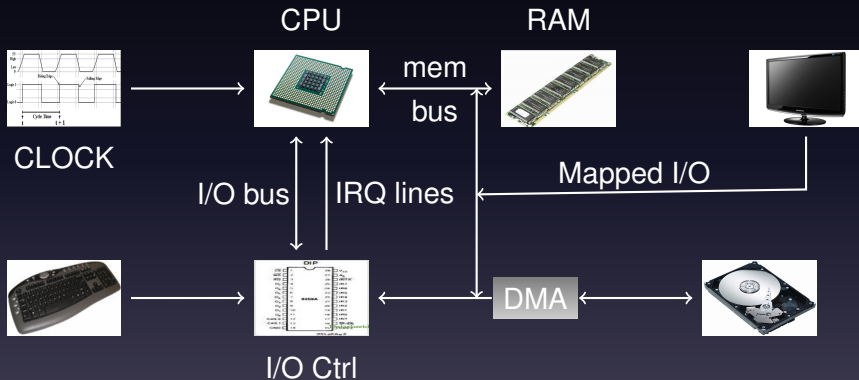
- **Tiempo compartido (timesharing)**

- Extensión de la multiprogramación para aplicaciones interactivas
- Previene monopolización de la CPU: uso por ráfagas (time-slice)

Evolución de UNIX



Hardware (arquitectura)



¿Qué es un Sistema Operativo?

- Un administrador de los recursos del sistema:
Uso compartido de CPUs, gestión de memoria, gestión de la entrada-salida, protección, . . .

¿Qué es un Sistema Operativo?

- Un administrador de los recursos del sistema:
Uso compartido de CPUs, gestión de memoria, gestión de la entrada-salida, protección, . . .
- Una abstracción del hardware
Es posible verlo como una máquina virtual

¿Qué es un Sistema Operativo?

- Un administrador de los recursos del sistema:
Uso compartido de CPUs, gestión de memoria, gestión de la entrada-salida, protección, ...
- Una abstracción del hardware
Es posible verlo como una máquina virtual
- Un conjunto de servicios para los procesos de usuario
 - Rutinas comunes de control de dispositivos (device drivers)
 - Define una interface de servicios (llamadas al sistema) para los programas de usuario

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos
- Administrador de la memoria: gestión y confinamiento

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos
- Administrador de la memoria: gestión y confinamiento
- Subsistema de entrada/salida: device drivers, interrupciones, . . .

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos
- Administrador de la memoria: gestión y confinamiento
- Subsistema de entrada/salida: device drivers, interrupciones, ...
- Sistema de archivos
 - Persistencia: programas, datos, organización, ...

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos
- Administrador de la memoria: gestión y confinamiento
- Subsistema de entrada/salida: device drivers, interrupciones, ...
- Sistema de archivos
 - Persistencia: programas, datos, organización, ...
- Subsistema de red

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos
- Administrador de la memoria: gestión y confinamiento
- Subsistema de entrada/salida: device drivers, interrupciones, ...
- Sistema de archivos
 - Persistencia: programas, datos, organización, ...
- Subsistema de red
- Protección: entre procesos, memoria, usuarios, archivos, ...

Principales componentes

- Administrador de procesos
 - Creación y finalización de procesos (tareas)
 - Brinda mecanismos de Inter-Process Communication (IPC)
 - Mecanismos de acceso exclusivo a recursos
- Administrador de la memoria: gestión y confinamiento
- Subsistema de entrada/salida: device drivers, interrupciones, ...
- Sistema de archivos
 - Persistencia: programas, datos, organización, ...
- Subsistema de red
- Protección: entre procesos, memoria, usuarios, archivos, ...
- Interfaz a aplicaciones (programas de usuario): llamadas al sistema

Llamadas al sistema

System calls

- Servicios a los programas de usuario
- Implementación: uso de traps (software interrupts)

Categorías

- **Procesos:**
 - Creación (fork), destrucción (kill), información (getpid), ...
- **Gestión de memoria:**
 - Reserva (sbrk), liberación, ...
- **Archivos y directorios:**
 - open, mkdir, unlink, read, write, seek, close ...
- ***IPC (Inte-Process communication)***
 - pipes, lock, unlock, ...

Programas, Procesos y threads

- **Programa:** Archivo ejecutable
 - Contiene segmentos de código y datos (bss)
 - Metadatos (punto de entrada, referencias externas, etc)

Programas, Procesos y threads

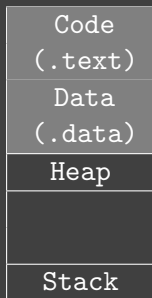
- **Programa:** Archivo ejecutable
 - Contiene segmentos de código y datos (bss)
 - Metadatos (punto de entrada, referencias externas, etc)
- **Proceso:** Instancia de un programa en ejecución
 - Imagen en memoria (bloques)
 - Código: instrucciones de máquina
 - Datos (bss): memoria estática para datos globales
 - Stack: pila de *registros de activación* (control+datos locales)
 - Heap: área usada para datos con tiempo de vida arbitrarios

Programas, Procesos y threads

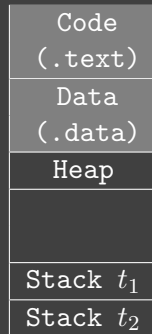
- **Programa:** Archivo ejecutable
 - Contiene segmentos de código y datos (bss)
 - Metadatos (punto de entrada, referencias externas, etc)
- **Proceso:** Instancia de un programa en ejecución
 - Imagen en memoria (bloques)
 - Código: instrucciones de máquina
 - Datos (bss): memoria estática para datos globales
 - Stack: pila de *registros de activación* (control+datos locales)
 - Heap: área usada para datos con tiempo de vida arbitrarios
- **Thread:** Hilo de un proceso
 - **Multithreading:** Múltiples hilos
 - Es necesario un stack por hilo
 - Puede hacerse en modo usuario (usando bibliotecas) o en modo kernel

Procesos y threads


Estructura de un proceso en memoria



Proceso



Proceso multithreaded

Los bloques  se cargan desde el archivo ejecutable.

Protección: modos de operación

- **Modo usuario:**

- No es posible ejecutar algunas instrucciones privilegiadas
 - Des/habilitar interrupciones
 - Instrucciones de I/O (ej: in, out en IA32)
 - Otras: (HALT, manipulación de registros especiales, etc)
- Procesos confinados en un área de memoria
 - Referencias a direcciones de memoria no permitidas disparan excepciones
 - Gestión de memoria a través de llamadas al sistema

Protección: modos de operación

- **Modo usuario:**

- No es posible ejecutar algunas instrucciones privilegiadas
 - Des/habilitar interrupciones
 - Instrucciones de I/O (ej: in, out en IA32)
 - Otras: (HALT, manipulación de registros especiales, etc)
- Procesos confinados en un área de memoria
 - Referencias a direcciones de memoria no permitidas disparan excepciones
 - Gestión de memoria a través de llamadas al sistema

- **Modo kernel:**

- No hay restricciones sobre el tipo de instrucciones a ejecutar
- Acceso a todos los recursos del sistema

Protección: modos de operación

- **Modo usuario:**

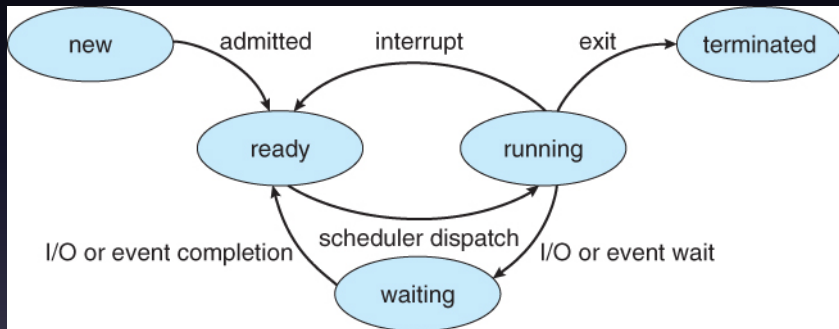
- No es posible ejecutar algunas instrucciones privilegiadas
 - Des/habilitar interrupciones
 - Instrucciones de I/O (ej: in, out en IA32)
 - Otras: (HALT, manipulación de registros especiales, etc)
- Procesos confinados en un área de memoria
 - Referencias a direcciones de memoria no permitidas disparan excepciones
 - Gestión de memoria a través de llamadas al sistema

- **Modo kernel:**

- No hay restricciones sobre el tipo de instrucciones a ejecutar
- Acceso a todos los recursos del sistema

- Pueden haber más de dos modos (anillos de protección):
Ej. 4 en x86

Control de procesos



Estados de un proceso.

Otros conceptos

- **Simmetric Multiprocessing (SMP):** multiprocesador donde son tratados en forma igualitaria

Otros conceptos

- **Simmetric Multiprocessing (SMP):** multiprocesador donde son tratados en forma igualitaria
- **ASimmetric Multiprocessing (AMP):** los procesadores pueden tener diferentes roles

Otros conceptos

- **Simmetric Multiprocessing (SMP):** multiprocesador donde son tratados en forma igualitaria
- **ASimmetric Multiprocessing (AMP):** los procesadores pueden tener diferentes roles
- **Uniform Memory Access (UMA):** el acceso a diferentes áreas de memoria tiene el mismo costo

Otros conceptos

- **Simmetric Multiprocessing (SMP):** multiprocesador donde son tratados en forma igualitaria
- **ASimmetric Multiprocessing (AMP):** los procesadores pueden tener diferentes roles
- **Uniform Memory Access (UMA):** el acceso a diferentes áreas de memoria tiene el mismo costo
- **Non-uniform Memory Access (NUMA):** el acceso a diferentes áreas de memoria tiene diferente costo
Ejemplo: sistemas con memoria local a cada procesador y acceso a una memoria global compartida

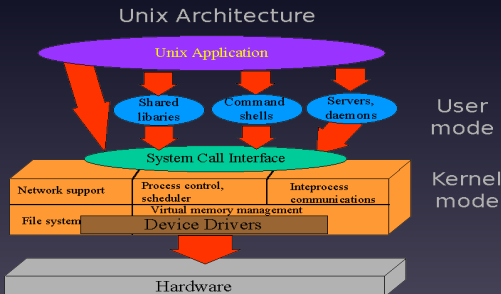
Otros conceptos

- **Simmetric Multiprocessing (SMP):** multiprocesador donde son tratados en forma igualitaria
- **ASimmetric Multiprocessing (AMP):** los procesadores pueden tener diferentes roles
- **Uniform Memory Access (UMA):** el acceso a diferentes áreas de memoria tiene el mismo costo
- **Non-uniform Memory Access (NUMA):** el acceso a diferentes áreas de memoria tiene diferente costo
Ejemplo: sistemas con memoria local a cada procesador y acceso a una memoria global compartida
- **Memoria caché:** Memoria de alta velocidad (atenuación de diferencias de velocidad entre la CPU y la RAM)
Puede haber varios niveles (ej: registros, L1, L2, L3, ...)

Estructura de Sistemas Operativos

- **Monolíticos**

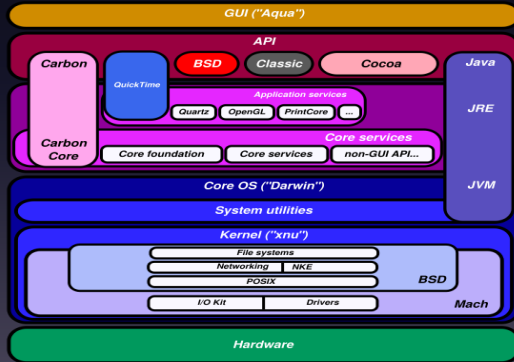
- Diferentes módulos del kernel en el mismo espacio de memoria
- Conceptualmente es un sistema concurrente con memoria compartida
- Ventajas: mayor rendimiento
- Desventajas: No hay protección entre subsistemas



Estructura de Sistemas Operativos

- **Microkernels**

- Protección entre módulos del kernel
- Comunicación por medio de ***pasaje de mensajes***
- Ventajas: Cada subsistema es un módulo de servicios
- Desventajas: menor performance (por copias de mensajes)

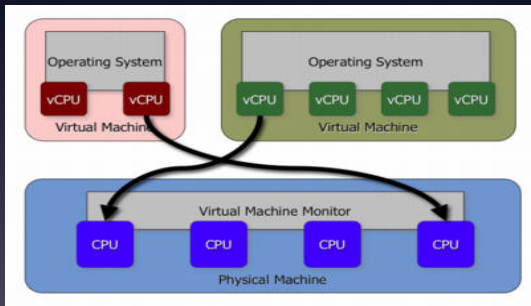


Arquitectura de MAC-OS

Estructura de Sistemas Operativos

- **Máquinas virtuales**

- Virtual machine monitor (hypervisor)
- Proveen una abstracción de hardware a diferentes SOs
- Ventajas: una única máquina puede correr varios SOs aislados
- Desventajas: difícil de implementar en algunas plataformas (ej: Intel's CPUs)



Llamadas al sistema (UNIX)

- **Creación de procesos:** *fork()*, *exit()* y *wait()*
fork() crea un proceso (hijo) idéntico al invocante (padre)
exit(code) finaliza un proceso (con código de salida *code*)
wait() espera a que finalice algún proceso hijo

```
void main(void) {  
    int pid = fork();  
  
    if ( pid == 0 )  
        exit(0);          /* in child: just finish... */  
    else  
        pid = wait();      /* in parent: wait for child exits */  
}
```

Llamadas al sistema (UNIX)

- **exec(cmd)** system call
reemplaza la imagen del invocante con el del programa *cmd*

```
void main(void) {  
    char * argv[3];  
  
    argv[0] = "echo";  
    argv[1] = "hello_world";  
    argv[2] = 0;  
    execve("/bin/echo", argv);  
}
```

Llamadas al sistema (UNIX)

- Ejecución concurrente (ej: *cmd1* & *cmd2*)

```
if ( fork() == 0 ) exec("cmd1" ,...);  
else {  
    if ( fork() == 0 ) exec("cmd2" ,...);  
    else wait();  
    wait();  
}
```

- Ejecución secuencial (ej: *cmd1* ; *cmd2*)

```
if ( fork() == 0 ) exec("cmd1" ,...);  
else wait();  
if ( fork() == 0 ) exec("cmd2" ,...);  
else wait();
```

Llamadas al sistema (UNIX)

- Archivos y descriptores:
 - En UNIX un *dispositivo* es un *archivo*.
 - `open()` retorna un *descriptor* de archivo.
 - Un proceso hijo (ver `fork()`) hereda de su padre los descriptores de archivos abiertos.
 - `init` inicialmente abre los descriptores 0 (entrada), 1 y 2 (salida y salida de errores estándar), asociados inicialmente al teclado y a la pantalla o *consola*, respectivamente.

```
/* simple cat */  
void main(void) {  
    char buf[512];  
    int n;  
  
    while ( (n = read(0, buf, sizeof(buf))) > 0 )  
        write(1, buf, n);  
    exit(0);  
}
```


Llamadas al sistema (UNIX)

- Redirección

Un *descriptor* puede redirigirse a otro archivo. Los operadores de redirección del shell son: `<` (entrada estándar), `>` (salida estándar), `2 >` (salida de errores). Ejemplo: el comando `cat < input.txt` es ejecutado por el shell de la siguiente forma:

```
...
char * argv[2]; int fd;

argv[0] = "cat";
argv[1] = 0;
if ( fork() == 0 ) {
    close(0);
    fd = open("input.txt", O_RDONLY);
    dup(fd); // standard input = fd (ver man dup)
    execve("cat", argv); // la entrada de cat es input.txt
}
...
```

Llamadas al sistema (UNIX)

- Pipes

Un pipe o tubo (|) es un buffer en el kernel expuesto a los procesos por un par de descriptores, (lectura y escritura).

```
/* Execute wc which read from pipe.  
   Parent process write to pipe. */
```

```
...
```

```
int p[2];  
  
pipe(p);  
if ( fork() == 0 ) {  
    close(0);  
    dup(p[0]);    /* desc. 0 = p[0] */  
    close(p[0]); close(p[1]);  
    execve("/bin/wc", argv);  
} else {  
    write(p[1], "hello_world\n", 12);  
    close(p[1]); close(p[0]);  
}
```

Llamadas al sistema (UNIX)

Sistemas de archivos

- Un archivo es una secuencia de bytes almacenados en bloques de tamaño fijo (no necesariamente contiguos) en un dispositivo de almacenamiento (ej: discos).
- Los *archivos* se organizan (pertenecen a) *directorios*.
- *Directorio*: contenedor de archivos. Llamadas al sistema: `chdir`, `mkdir`, `rmdir`, `rm`, `unlink` (borrado), ...
- Un *directorio* es un archivo que contiene una tabla de *metadatos* (nombre y atributos) de los archivos que contiene.

Llamadas al sistema (UNIX)

Sistemas de archivos (cont.)

- Enlaces (links): *alias* de archivos o directorios (ver comando `ln`).
- Utilidad: Simplificar el *camino* (*path*) a un archivo.
- Tipos de enlaces:
 - 1 *Simbólicos*: Si se eliminan no se elimina el original.
 - 2 *Duros* (*hard*): Eliminar el enlace también elimina el original.

Las llamadas al sistema sobre enlaces (`ln`, `ln -s`) y `unlink` (eliminación de archivos) sólo agregan/eliminan entradas en directorios.