NSO Modulo 1 version 2010

Sistemas Operativos (Universidad Nacional de La Matanza)

Módulo 1

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

CONTENIDO:

- Conceptos introductorios a los S.O.
- Definición y Funciones.
- Características de Hardware necesario.
- Características de los S.O.
- Características comunes a todos los S.O.
- Componentes.
- Propósitos.
- Distintos Tipos de S.O.
- Conceptos básicos sobre Trabajos y Procesos.
- Prestaciones de un S.O.
- Servicios prestados por el S.O.
- S.O. para multiprocesadores.

OBJETIVOS DEL MODULO: Dar los conceptos básicos sobre las funciones; composición, arquitecturas, módulos y servicios, de los Sistemas Operativos Computacionales en general.

OBJETIVOS DEL APRENDIZAJE: Después de la lectura y del estudio del presente módulo el alumno conocerá:

(1) Los conceptos y elementos componentes de los Sistemas Operativos.
(2) Los distintos tipos y estructuras de los Sistemas Operativos.
(3) Concepto de ambiente de ejecución y de cambio de contexto.
(4) Conceptos básicos sobre los Sistemas Operativos, sus propósitos, los servicios que brinda y su ejecución.
(5) Conocer la terminología y sus significados utilizados en éste módulo.
(6) Conocer los Sistemas Operativos para multiprocesadores.

Metas del Módulo 1

En éste módulo se introducirá los conceptos necesarios e indispensables sobre un Sistema Operativo, como ser: qué es un Núcleo (Kernel)[1] o interfase con el Hardware, la interfase con el usuario (Shell), las características del Hardware que requiere un S.O. para su ejecución, el concepto de arquitectura de Sistemas Operativos, que es un proceso, hilos, etc.; para comprender con mayor precisión como hace ejecutar los programas en un computador.

OBSERVACIÓN: Usaremos las siglas S.O. o SO indistintamente para referirnos al término Sistema Operativo.

Requerimientos previos del Módulo 1

Para estudiar y comprender este módulo, es necesario, primero, repasar los conceptos que se vieron en la asignatura previa de Arquitecturas de computadores y que se acompaña una breve

[1] **Nota Importante**: Los términos específicos de la Asignatura se escribirán en Castellano y en Ingles y constituyen el glosario de la materia y se usarán indistintamente.

descripción como el anexo A de este libro. Segundo, también es recomendable estudiar los restantes Anexos dado que los contenidos son necesario para comprender los módulos que se irán estudiando durante este curso.

Como podrá observarse en la Bibliografía de cada módulo, se ha consultado todo tipo de libros sobre Sistemas Operativos y desde distintos puntos de vista, diversos sitios en Internet que colaboraron con una amplia gama de imágenes y conceptos.

1.1 Introducción a los Sistemas Operativos.
a) Introducción a los Sistemas Operativos. b) Conceptos previos sobre los Sistemas Operativos

a) Introducción a los Sistemas Operativos.

A finales de los 40's el uso de computadoras estaba restringido a aquellas empresas o instituciones que podían pagar su alto precio, y no existían los *sistemas operativos*. En su lugar, el programador debía tener un conocimiento y contacto profundo con el *hardware (Compuesto por Válvulas)*, y en el infortunado caso de que su programa fallara, debía examinar los valores de los registros y paneles de luces indicadoras del estado de la computadora para determinar la causa del fallo y poder corregir su programa, además de enfrentarse nuevamente a los procedimientos de aportar tiempo del sistema y poner a punto los compiladores, vincuadores o editores de enlaces (linkers), etc.; para volver a correr su *programa*, es decir, enfrentaba el problema del *procesamiento en serie o secuencial* (serial processing).

La importancia de los *sistemas operativos* nace históricamente desde los años 50's, cuando se hizo evidente que el operar una computadora por medio de tableros enchufables (*primera generación),* y luego por medio del trabajo en lote (*segunda generación -se reemplaza la válvula por el transistor-),* podía mejorarse notoriamente, pues el operador realizaba siempre una secuencia de pasos repetitivos, lo cual es una de las características contempladas en la definición de lo que es un *programa*. Es decir, se comenzó a ver que las tareas mismas del operador podían plasmarse en un programa, el cual a través del tiempo y por su enorme complejidad se le llamó "*Sistema Operativo*" *(Operating System*). Así, tenemos entre los primeros sistemas operativos al Fortran Monitor System (FMS), e IBMSYS.

Posteriormente, en la *tercera generación* de computadoras nace uno de los primeros sistemas operativos con la filosofía de administrar una familia de computadoras: el OS/360 de IBM. Fue este un proyecto tan novedoso y ambicioso que enfrentó por primera vez una serie de problemas conflictivos debido a que anteriormente las computadoras eran creadas para dos *propósitos* en general: el comercial y el científico. Así, al tratar de crear un solo sistema operativo para computadoras que podían dedicarse a uno de los propósitos, al otro o a ambos, puso en evidencia la problemática del trabajo en equipos de análisis, diseño e implantación de sistemas grandes. El resultado fue un sistema del cual uno de sus diseñadores expresó su opinión en la portada de un libro: *una horda de bestias prehistóricas atascadas en un foso de brea.*

Surge también en la *tercera generación* de computadoras el concepto de la *multiprogramación*, porque debido al alto costo de las computadoras era necesario idear un esquema de trabajo que mantuviese a la *unidad central de procesamiento (CPU - Central Processing Unit)*, más tiempo ocupada, así como el *encolado (spooling)*, *de trabajos* para su lectura hacia los lugares libres de memoria o la escritura de resultados. Sin embargo, se puede afirmar que los sistemas durante la tercera generación siguieron siendo básicamente sistemas de lotes (*Batch Processing*).

En la *cuarta generación* la electrónica avanza hacia la integración a gran escala (chips), pudiendo crear circuitos con miles de transistores en un centímetro cuadrado de silicio y ya es posible hablar de las *computadoras personales* y las *estaciones de trabajo*. Surgen los conceptos de *interfases amigables* intentando así atraer al público en general al uso de las computadoras como herramientas cotidianas. Se hacen populares los sistemas operativos: el MS-DOS y UNIX en estas máquinas. También es común encontrar *clones* de computadoras personales y una multitud de empresas pequeñas ensamblándolas por todo el mundo.

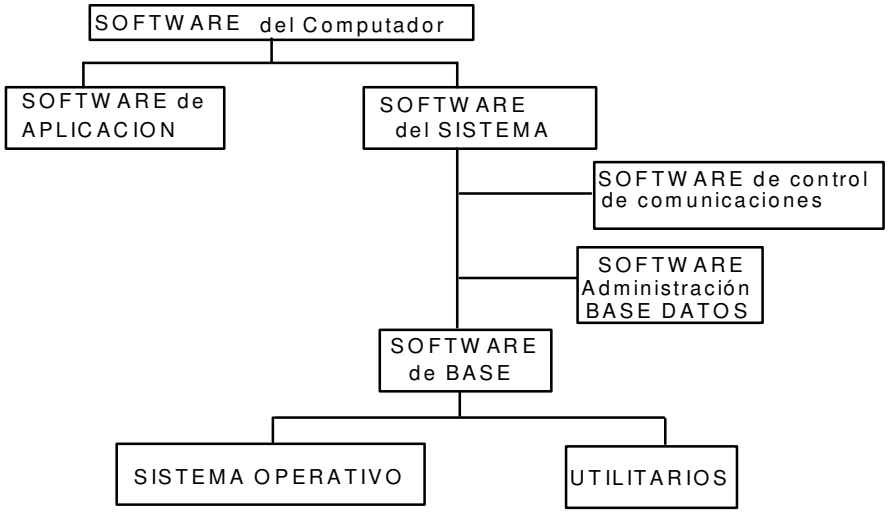


Fig. 1.01. Distintos tipos de software de un computador

Para mediados de los 80's, comienza el auge de las *redes de computadoras* y la necesidad de *sistemas operativos en red* y *sistemas operativos distribuidos*. La red mundial Internet se va haciendo accesible a toda clase de instituciones y se comienzan a dar muchas soluciones (y problemas), al querer hacer convivir recursos residentes en computadoras con sistemas operativos diferentes. Para los 90's el *paradigma de la programación orientada a objetos* cobra auge, así como el manejo de objetos desde los sistemas operativos. Las aplicaciones intentan crearse para ser ejecutadas en una plataforma específica y poder ver sus resultados en la pantalla (o monitor) de otra diferente (por ejemplo, ejecutar una simulación en una máquina con UNIX y ver los resultados en otra con DOS). Los *niveles de interacción* se van haciendo cada vez más profundos.

Sin el *software,* una computadora (ver Fig. 1.01), es en esencia una masa metálica sin utilidad. Con el software, una computadora puede almacenar, procesar y recuperar información, encontrar errores de ortografía e intervenir en muchas otras valiosas actividades. El *software* para computadoras puede clasificarse en general, en 2 clases: los *programas de sistema*, que controlan la operación de la computadora en sí y los *programas de aplicación*, los cuales resuelven problemas de los usuarios. El *programa fundamental* de todos los programas de sistema, es el *Sistema Operativo*, que controla todos los recursos de la computadora y proporciona la base sobre la cual pueden escribirse los programas de aplicación. Los otros programas son los Administradores de Bases de Datos y el de control de las comunicaciones.

Un ***Sistema Operativo* es un programa** que debe estar instalado en un computador y que actúa como intermediario entre el usuario y el hardware de un computador y su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar sus programas. El objetivo principal de un Sistema Operativo es, entonces, lograr que el Sistema de computación se use de manera cómoda, y el objetivo secundario es que el hardware del computador se emplee de manera eficiente.

b) Conceptos sobre los Sistemas Operativos

En cuanto a los Sistemas Operativos podemos afirmar que:

- Depende del Hardware que controla debido a que complementa al juego de instrucciones del procesador.
- Puede ser ejecutado por el mismo procesador que ejecuta el programa del usuario.
- Cuando se está ejecutando el programa usuario, el S.O. está inactivo si es monoprocesador.
- Los detalles de interfases internas varían de máquina en máquina, dado que está

íntimamente ligado a los recursos del Hardware, que administra y controla.

- Las capacidades externas y la interfase humana varían en función de las prestaciones que ofrece el S.O. Muchas veces sirven como argumentos de venta o atracción tecnológica sin ofrecer substanciales mejoras en los servicios que dice haber incorporado el fabricante o proveedor del Sistema Operativo.
- Los servicios que provee el S.O. a los programas usuarios cuando estos se ejecutan, son denominados llamadas al sistema (System Calls o SYSCALL).

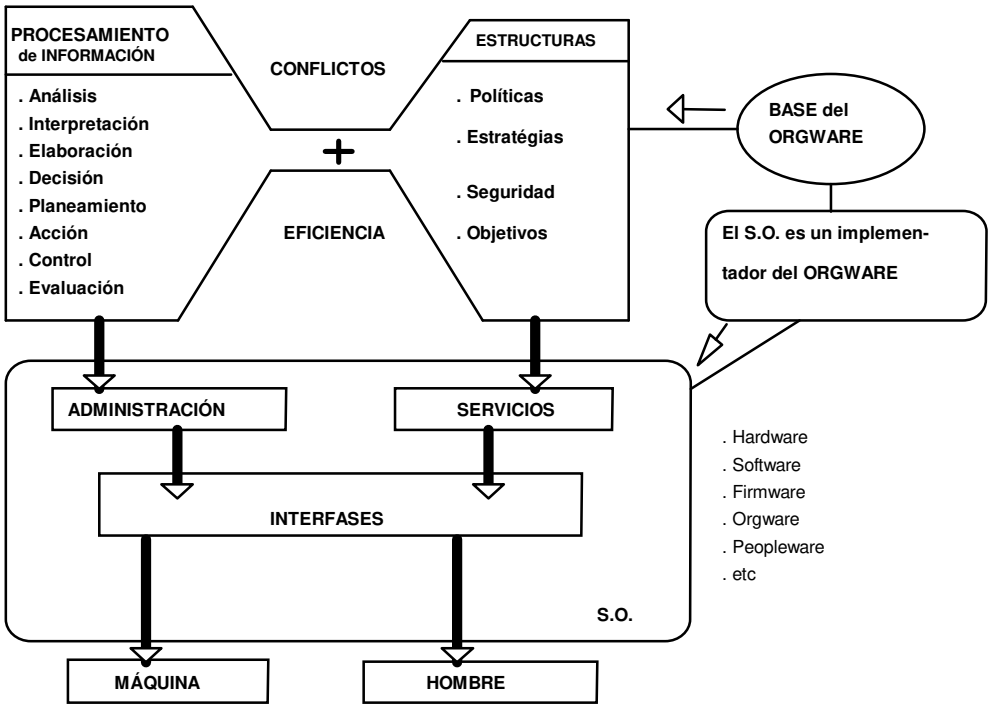


Fig. 1.02 Esquema de interrelaciones de un sistema operativo.

1.1.1. Sistema de Procesamiento de Datos y sistema operativo: a) Componentes de un sistema informático. Sistema lógico, sistema físico y sistema humano. b) Aplicaciones y programas del sistema. c) Definiciones y Concepto de sistema operativo.

La INFORMÁTICA es la disciplina que estudia el tratamiento automatizado de la información, incluyendo como aspectos más relevantes:

- El diseño de las computadoras.
- La programación de las computadoras.
- El procesamiento de la información.
- La resolución de problemas mediante algoritmos.
- El estudio de los algoritmos en sí mismos.

a) Componentes de un sistema informático: sistema lógico, sistema físico y sistema humano.

Entre algunas definiciones informales podemos encontrar:

Un ***computador*** es una máquina capaz de ejecutar instrucciones sobre datos (que pueden ser proporcionados externamente), permitiendo la consulta de los resultados. Las instrucciones se encuentran en algún soporte modificable.

Un ***algoritmo*** es una especificación más o menos detallada y no ambigua de la secuencia de pasos a seguir para resolver un determinado problema con un determinado lenguaje de programación ejecutándose en un computador.

A la parte física que trata la informática se la denomina ***hardware.*** A la parte lógica que trata la informática se la denomina ***software***. La aplicabilidad de la informática es prácticamente universal.

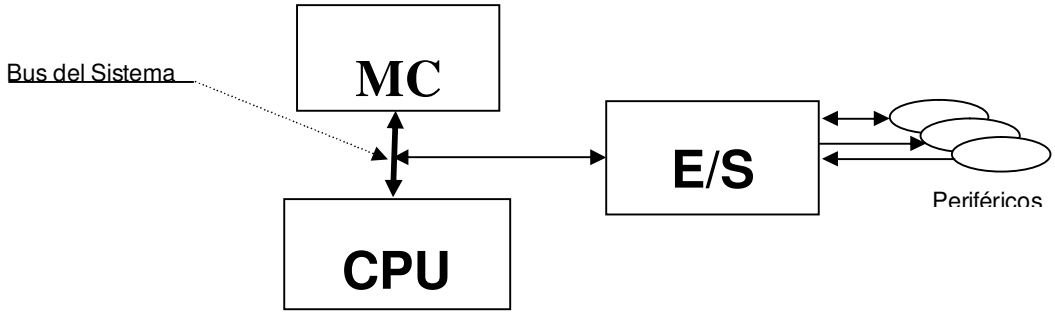


Fig. 1.03 Principales módulos de un computador

EL HARDWARE (HW) DE UN COMPUTADOR se compone de:

- **Un Sistema Central** (ya sea un Mainframe, o uno o mas servidores (servers) en una Red).
- **Unidad Central de Proceso** (CPU): encargada de ejecutar las instrucciones de los programas. Puede haber una o más por Sistema también llamado **procesador**.
- **Memoria Central** (MC)[2]: encargada de almacenar tanto las instrucciones de los programas como también los datos y los resultados.
- **Unidades de Entrada y Salida** (E/S): que permite conectar los dispositivos periféricos al bus del Sistema (Utilizaremos las siglas E/S o I/O como sinónimo de Entrada-Salida).
- **Un conjunto de Dispositivos Periféricos:** que ofician de Comunicación y los intercambios de datos con el exterior.
- **Dispositivos de Almacenamiento Secundario**: para mantener información organizada y guardada en un soporte en forma permanente.
- **Medios de comunicación**: entre los diferentes componentes (cableado o *buses*).

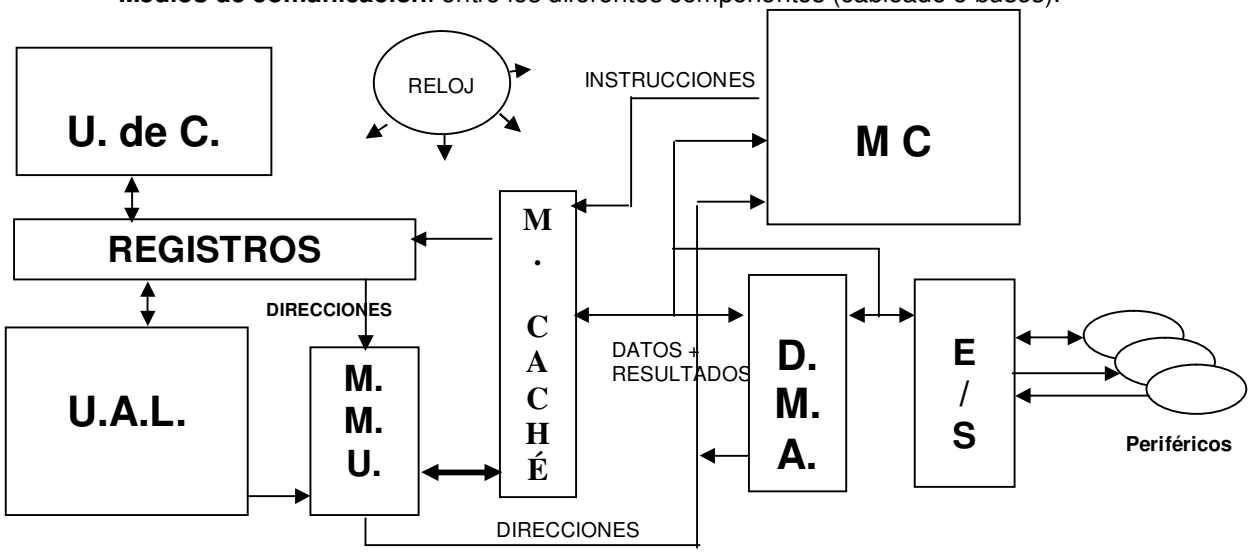


Fig. 1.04 Interrelación de módulos funcionales de un computador

LA UNIDAD CENTRAL DE PROCESO (CPU o Procesador), se compone de:

- **Reloj:** generador de señales temporizadas que marcan las fases en la ejecución de una instrucción del procesador. El período de la señal producida por el reloj se denomina ciclo y

[2] Usaremos en este texto el término Memoria Central (MC), o Memoria Principal (MP), para indicar que se trata del almacenamiento físico que se accede desde el registro Program Counter (PC), de la CPU.

se mide en Mega Hertz (MHz - Millones de ciclos por segundo), o Giga Hertz (GHz - mil millones de ciclos por segundo). El procesador efectúa acciones que tienen una duración expresada en múltiplos de tales ciclos. En la mayoría de las ocasiones se encuentra (físicamente), fuera del propio procesador.

- **UNIDAD DE CONTROL** (UC): Extrae de la MC la nueva instrucción a ejecutar. Analiza dicha instrucción y establece las conexiones eléctricas correspondientes dentro de la UAL. (Unidad Aritmética Lógica) u otros módulos funcionales. Además, extrae de la MC los datos implicados en la instrucción. Desencadena el tratamiento de dichos datos en la UAL u otros módulos funcionales. Almacena los resultados (si los hubiera), en la MC.
  Las operaciones de acceso a MC se realizan con una cantidad de información denominada palabra o word (su tamaño depende de cada procesador). En ocasiones es necesario realizar varios accesos para obtener o depositar una información completa en los registros de la CPU.
- **UNIDAD ARITMÉTICO-LÓGICA** (**UAL**): opera con los datos según indicaciones de la UC.
- **REGISTROS:** almacenamiento temporal de información para su procesamiento.

La CPU es el elemento más rápido de todo el computador (de unidades a centenares de millones de instrucciones por segundo).

LA MEMORIA CENTRAL (MC) almacena dos tipos de información:

- **Instrucciones:** es decir, informaciones que indican qué operaciones se efectuarán en el procesador con los datos
- **Datos:** que se transformarán en otros datos o resultados después que las instrucciones lo tratan.

Sin embargo, ambos tipos de información no se distinguen externamente en nada: será la UC la que decida que una información se interprete como instrucción o como dato.

La máquina puede ejecutar un programa inicialmente almacenado en la MC. El programa actúa, a través del procesador, sobre los datos en la MC y almacena los resultados en la MC a medida que se vayan obteniendo del procesamiento.

El mecanismo para acceder a la información almacenada en MC es el siguiente: La información está organizada en palabras. Cada palabra tiene una dirección asociada.

El procesador genera una dirección para acceder a la información deseada mediante un registro puntero llamado Program Counter (PC).

La MC selecciona de entre todas sus informaciones la que tenga asociada tal dirección.

Se efectúa la operación apropiada de Lectura o Escritura.

- Lectura: la MC le devuelve al procesador la información (palabra), contenida en la dirección especificada que fuera cargada en el Program Counter.
- Escritura: el procesador ordena a la MC que se deposite una información en la dirección especificada.

La información almacenada en la MC se expresa en binario.

Un bit (*BInary unit*): es una cifra con dos valores posibles: 0 y 1. Su capacidad de expresión es muy reducida (SI / NO, VERDADERO / FALSO, BLANCO / NEGRO, etc.). Los bits pueden agruparse para aumentar su capacidad de representación. Con 2 bits hay 4 combinaciones posibles (00, 01, 10, 11), que podrían representar a cuatro letras o números, por ejemplo.

En general, con N bits se pueden representar combinaciones distintas, y a cada una se le puede asignar una interpretación diferente.

Una agrupación habitual es el ***byte (binary termin)****,* término binario que generalmente contiene 8 bits, pero puede ser de ***n*** bits dado que es una convención. Con 8 bits, permite 256 combinaciones diferentes (2 elevado a la 8va. Potencia). Es útil para representar un carácter (alfabeto en minúsculas y mayúsculas, signos de puntuación, dígitos del 0 al 9, caracteres especiales, etc.).

Existe una tabla que relaciona combinaciones de 7 bits con los caracteres que representan. Esta tabla utiliza el código ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange), de 7 bits. (para completar el byte, reserva un bit para detección de errores por lo que sólo representa sumatorias de bits par o impar). También hay otro código llamado ASCII Extendido de 8 bits que contiene el de 7 bits y no el de paridad.

Magnitudes: se utiliza el sistema binario[3].

- **1 Kibibit** = 1024 bits. Aplicado a bytes, representa aproximadamente la información ocupada por media página de texto.
- **1Mebibit** = 1.048.576. Aplicado a bytes, viene a representar la información ocupada por un libro (de unas 500 páginas).
- **1Gibibyte** = 1.073.741.824 By. Representa la información de unos 1000 libros.
- **1Tebibyte** = Terabyte binarios (Tby) (2^{12})
- **Pebibyte** = Peta byte binario (Pby) (2^{15})
- **Exbibyte** = Exa byte binario (Eby) (2^{18})
- **Zebibyte** = Zetta byte binario (Zby) (2^{21})
- **Yobibyte** = Yotta byte binario (Yby) (2^{24})

Algunas características de la Memoria Central son:

- Tamaño reducido (orden de Mebibytes o algunos Gibibit).
- Tiempo de acceso pequeño (orden de los microsegundos o nanosegundos), y constante.
- Acceso directo (cada posición de memoria tiene una dirección).

Tipos de memoria:

- **RAM**: Memoria de lectura y escritura. En algunos casos es Volátil (al apagar el computador, la información se pierde) en otros no (Flash RAM). Útil para instrucciones y datos (es la memoria normal).
- **ROM**: Memoria de sólo lectura. Permanente se utiliza especialmente en el arranque del computador, conteniendo las primeras instrucciones del programa de inicialización.

Los siguientes términos significan:

- **Hardware:** (literalmente: material duro), es todo lo referente a la computadora que puede ser "tocado", o sea, lo físico. Por Ejemplo: teclado**,** monitor, circuitos, chips, plástico, hierro, etc.
- **Software**: (literalmente: material "blando"), es todo lo lógico, Por Ejemplo: todo lo referente a Programas y a la gestión del computador.
- **Firmware**: es un programa grabado sobre un componente del sistema y que está integrado a él en funcionamiento.

El componente humano de un sistema informático esta relacionado con:

- **Orgware:** se refiere a todo lo estructurado y normado dentro de la organización, ya sea lo lógico, o lo físico y su uso.
- **Peopleware**: todo lo relacionado con los usuarios.

Existen distintos tipos de usuarios en función de los privilegios que se le asigna en el sistema, así por ejemplo, el ***superusuario (root, super user o system manager)***, dispone de todos los privilegios, lo cual le permite administrar sin restricciones el sistema mientras los otros usuarios solo tienen parte de los privilegios de acuerdo a sus funciones.

b) Aplicaciones y programas del sistema.

TRABAJO (JOB): es un conjunto de programas (generalmente de aplicaciones) y datos de los cuales se quiere obtener un resultado mediante las ejecuciones en un computador.

Un trabajo puede dividirse en ***pasos de trabajo (job steps)***, que son los procedimientos o funciones incorporados por el programador en el programa. Cada uno de estos pasos es una unidad de trabajo que se hace en forma secuencial mediante procesos (por ejemplo el ensamblado de un programa), una vez que el S.O. ha aceptado el Trabajo. Entonces podemos definir a un ***proceso (process)***, *como un conjunto de operaciones que se ejecutan en forma secuencial en el tiempo.*

Los procesos pueden ejecutarse en forma concurrente entre si. El conjunto de programas y datos

[3] **COMING TO TERMS WITH BYTES**
Computer terminology is becoming more precise: the International Electrotechnical Commission (IEC), which creates standards for electronic technologies, is adopting new prefixes to describe data values. The new term "kibibyte" will more accurately describe the number of bytes in a kilobyte -- rather than being 1,000, as could be inferred by the prefix "kilo," a kilobyte actually has 1,024 (2 to the 10th power) bytes. The metric prefixes currently employed -- kilo, mega, giga, etc. -- accumulate as a power of 10, rather than the binary system used in computer code. Thus, the Commission will use kibi, mebi, gibi, tebi , pebi and exbi to express exponentially increasing binary multiples (2 to the 10th power, 2 to the 20th power, etc.). "There was a need to straighten this out," says Barry Taylor of the National Institute of Standards and Technology.
(Science 12 Mar 99)

accedidos por el procesador durante un proceso se denomina ***espacio de direccionamiento (address space)***, de dicho proceso.

El S.O. vincula el espacio de direccionamiento de los procesos en la Memoria central con la máquina real en el momento de ejecución. Esto es la ubicación de dichas instrucciones en la memoria física realizadas por hardware en la mayoría de los casos. La ejecución significa que se suceden los procesos del usuario mezclados con las ***llamadas al sistema (System Calls***), conmutándose el contexto de ejecución en cada cambio. Este cambio de contexto se efectúa por "***sincronización",*** o por "***códigos puros" (pure code***), es decir, que no se modifican a sí mismos.

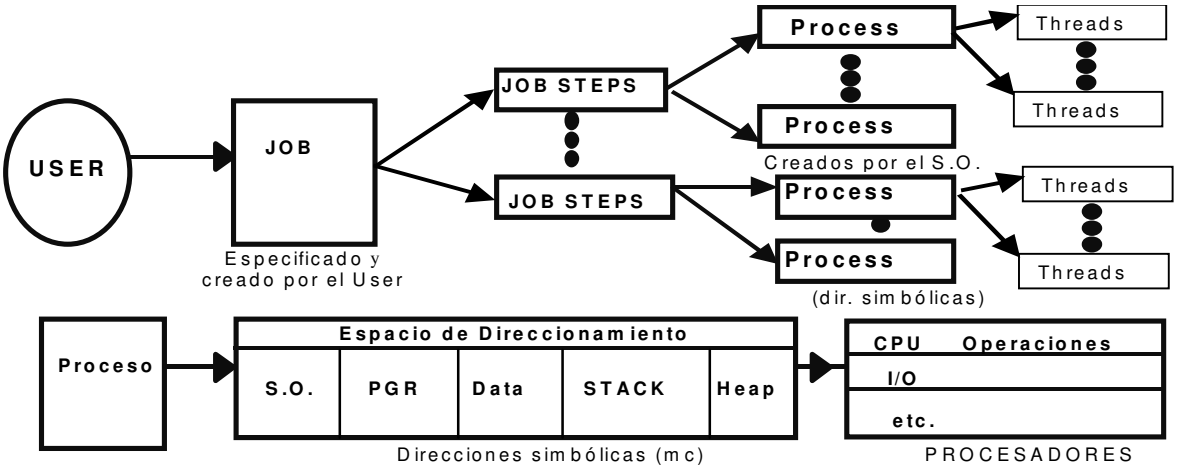


Fig. 1.05 Representación de los pasos desde Trabajo a Procesos y su espacio de direccionamiento

Resumiendo:

- **usuario** (*user*): *Es aquel que somete un trabajo (Job), a un sistema de cómputos*.
- **trabajo** (*Job*): *Conjunto de programas y datos sometidos al Sistema Operativo*.
- **paso de trabajo** (*Job Steps*): *Unidad del Job que debe ejecutarse en forma secuencial mediante una serie de procesos o tareas.*
- **proceso** (*Process*): *Sucesión de operaciones que se ejecutan secuencialmente en el tiempo* (luego, en el Módulo 2 se ampliará este concepto).
- **Threads o Hilos:** Procesos livianos que se ejecutan sobre una tarea.

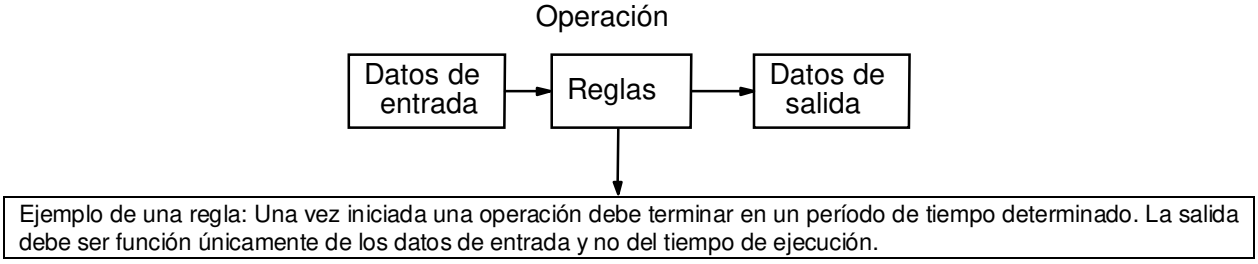


Figura 1.06 Esquema de procesamiento de datos.

Sincronización: Ordenamiento expreso en el tiempo de eventos u operaciones entre procesos o entre procesos y el S.O.

Multiprogramación (multiprogramming o multitasking): La multiprogramación se refiere a varios ***programas activos***, residentes en una memoria central ejecutándose sobre una CPU, mientras que ***multitarea*** significa que se ejecutan varias tareas "simultáneamente".

Es un multiplexado de los recursos (en especial la CPU), para crear la ilusión de que varios programas se ejecutan en forma simultánea.

Multiprocesamiento (multiprocessing): significa que se usan varios procesadores para ejecutar los programas.

LOS PROGRAMAS (Software):

Un programa se puede comparar con una lista de instrucciones. En lugar de llevarla a cabo una persona, lo hace una máquina.

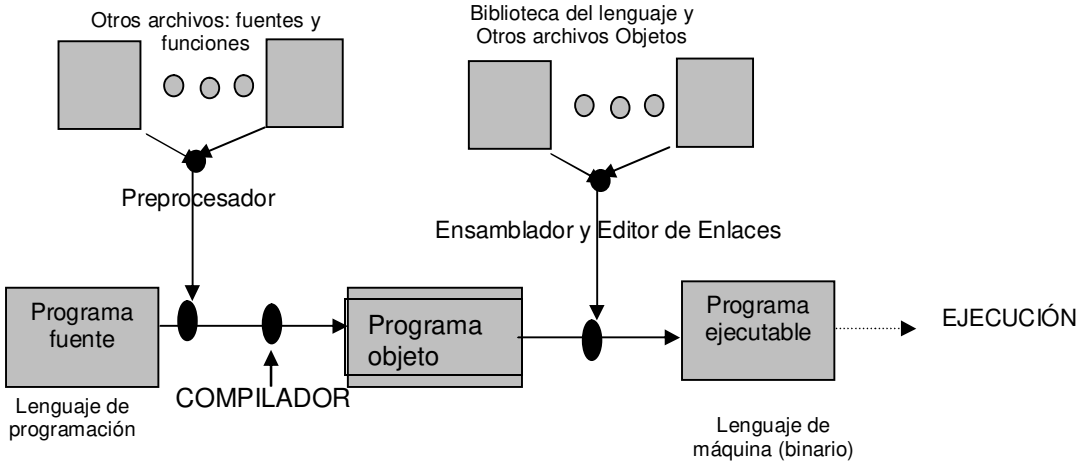


Fig. 1.07 Preparación de un Programa para su ejecución

Dado que el nivel de conocimiento del computador es muy reducido comparativamente con el ser humano, las instrucciones que puede ejecutar son muy sencillas (aunque a un ritmo muy elevado), y por lo tanto también lo es el lenguaje que puede entender. Esto lleva que programas auxiliares traduzcan la especificación producida por el programador en otro programa mucho más detallado que el computador puede entender y ejecutar en lenguaje binario. Para poder efectuar esta conversión, el programador está sujeto a un conjunto de reglas de sintaxis que debe respetar; y para cada instrucción codificada se asocia una semántica que el programador debe conocer.

El conjunto de sintaxis y semántica define un lenguaje de programación, y el programa que detalla esas instrucciones en algo que pueda ser entendido por la máquina se denomina traductor (interprete o compilador).

Habitualmente, el tipo de traductor usado es un compilador. Éste traduce el programa original (fuente), en un programa intermedio (objeto), que puede ser combinado con otros programas objeto (en ese caso cada uno es un fragmento incompleto), para dar lugar, al final, un programa ejecutable. En cualquier caso, obsérvese que esta es una visión aproximada y deficiente de cómo se lleva a cabo todo el proceso.

TIPOS DE PROGRAMAS

Virtualmente hay tantos tipos de programas como temas a los que se puede aplicar la informática (concretamente la programación); no obstante, y aunque con el planteamiento anterior la variedad es casi ilimitada, podemos destacar los siguientes tipos de programas:

Programas de uso universal: los que se ocupan de tareas comunes, como el tratamiento de información de naturaleza:

- **Numérica**: La contabilidad personal o de una empresa puede ser mantenida adecuadamente con una hoja de cálculo.
- **Textual**: Utilizando al computador y a los programas editores/procesadores de texto como sustitutos avanzados de la máquina de escribir.
- **Gráfica**: Dejando de lado, o complementado, el material de dibujo, tanto artístico (programas de dibujo libre), como técnico (programas de dibujo vectorial y CAD).
- **Variada pero organizada**: Con un sistema de gestión de bases de datos, la información (independientemente de su volumen), está siempre accesible, puede ser modificada calculadamente, y puede ser reorganizada por el criterio que parezca más conveniente.
- **Información "a distancia"**: Actualmente existen líneas de comunicación con suficiente capacidad para que el intercambio de información sea frecuente entre computadoras o personas distantes. En este campo pueden clasificarse los programas de mensajería electrónica (o correo electrónico), y los de transferencia de información (por ejemplo a través de módem).

Programas de uso localizado: los que son habituales sólo dentro de alguna ocupación o disciplina. Ejemplo Aplicaciones jurídicas, o en medicina.

Programas Utilitarios: Todos los que ayudan a la creación y verificación de los programas, como los traductores (compiladores e intérpretes), depuradores (Debuggers), etc ..

Esta clasificación no pretende ser exhaustiva, pero ilustra algunas facetas que pueden ser automatizadas a través de un computador dentro del trabajo normal y del especializado.

c) Definiciones de Sistema Operativo

Con los conceptos ya vistos en asignaturas previas, sobre procesamiento de la información al cual le sumamos un conjunto de estructuras, como: políticas, estratégicas, seguridad y objetivos; más los conflictos que deben ser resueltos con eficiencia; podemos definir el término Sistema Operativo.

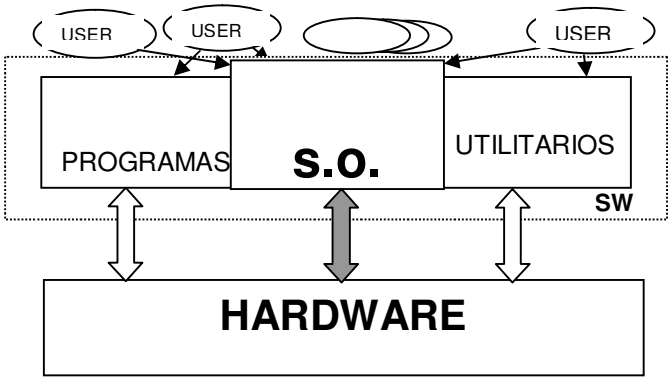


Fig. 1.08 Interrelación sel Software (SW), con el Hardware (HW).

Se pueden imaginar un Sistema Operativo como los programas, instalados en un sistema computacional o en un firmware, que hacen utilizable al hardware. El hardware proporciona la "capacidad bruta de cómputo"; los sistemas operativos ponen dicha capacidad de cómputo al alcance de los usuarios o programas mediante servicios y administran cuidadosamente los recursos del sistema para lograr un buen rendimiento.

Los Sistemas Operativos son básicamente administradores de recursos. El principal recurso que administran es el hardware del computador; además de los procesadores, los medios de almacenamiento, los dispositivos de entrada/salida, los dispositivos de comunicación, administran los procesos, el procesamiento y los datos.

Un Sistema Operativo es un programa que actúa como intermediario o interfase entre el usuario y el hardware del computador y su propósito es proporcionar el entorno en el cual el usuario pueda ejecutar programas. Entonces, el objetivo principal de un Sistema Operativo es, lograr que el sistema de computación se use de manera adecuada y *cómoda*, y el objetivo secundario es que el hardware del computador se emplee de manera *eficiente*.

Existen diversas definiciones de lo que es un Sistema Operativo, pero no hay una definición exacta, es decir una que sea estándar; a continuación se presentan algunas:

Sistema Operativo es un conjunto de módulos o funciones (software), que instalados en el computador, se ocupan de controlar y administrar la ejecución de los programas sobre los recursos que brinda el equipo (hardware), tales como: memoria, procesador, periféricos, etc. más otras funciones específicas.

Entonces, el S.O. tiene como principal objetivo incrementar la productividad de usuarios y el sistema de procesamiento.

Podemos afirmar que es el conjunto de programas que **residen** en la memoria central y que permite gestionar las diversas tareas de un computador, descargando a los usuarios de toda programación rutinaria y facilitando a cada uno de ellos compartir el uso de la máquina.

En esta afirmación la palabra compartir tiene un significado específico: los usuarios "compiten"

por el uso de los recursos físicos, tales como memoria, dispositivos, tiempo de uso del procesador, y cooperan con el uso de recursos lógicos: programas y datos. Esto es válido para un sistema multiusuario, como un equipo grande, también llamado Mainframe o HOST. En cambio en uno más chico, monousuario, como por ejemplo, un computador personal PC, generalmente no se comparten los recursos entre usuarios pero si se comparten entre los procesos del usuario.

Entonces, También podemos definir a un *Sistema Operativo* como un programa o conjunto de programas que reside en la Memoria Central de la máquina; y un conjunto de procedimientos (normas y operaciones), manuales y automáticos que hacen posible que la máquina funcione.

Resumiendo, Sistema Operativo es:

Un conjunto de programas que reside en la computadora y actúa como interfase entre los usuarios y los recursos, brindando los servicios necesarios para ejecutar programas.

Un conjunto de programas, rutinas y procedimientos manuales y automáticos que administran los recursos de un sistema de cómputos en operación para permitir a un grupo de usuarios, compartir eficientemente el mismo.

También lo podemos definir al sistema operativo por sus funciones:

Un programa o conjunto de programas residentes en la computadora responsable de las siguientes funciones:

- Inicializar la máquina: Chequeo de recursos y I.P.L. (Initial Program Loader), o bootstrapping (booteo en la jerga).
- Administrar los recursos del sistema.
- Servir de interfase hombre-máquina.
- Proteger a los usuarios y a los recursos

También podemos definirlo por el significado de sus palabras:

- **Sistema:** Conjunto de Personas, datos, software, comunicaciones, hardware y cosas que ordenadamente relacionados entre sí, contribuyen a lograr un determinado objetivo, en un ambiente.
- **Operativo**: personas, máquinas y cosas que trabajan conjuntamente y consiguen el objetivo deseado en ese ambiente.

SISTEMA OPERATIVO: Conjunto de Programas que ordenadamente relacionados entre sí, contribuyen a que la computadora lleve a cabo correctamente su trabajo para nosotros en un ambiente dado.

Habíamos dicho que un Sistema Operativo era un programa que actúa como interfase entre el usuario de una computadora y el hardware de la misma, entonces, el propósito es proveer un entorno en el cual el usuario puede ejecutar programas.

En realidad, un S.O. puede ser contemplado como una colección organizada de extensiones de software que se ejecuta sobre un Hardware determinado. Este Software consiste en rutinas que hacen funcionar el computador y proporcionan servicios y un entorno de ejecución para los programas.

Un S.O. NO es:

- **Un lenguaje o un compilador**: Existen S.O. que tienen integrado un lenguaje y su respectivo compilador con el que fueron construidos, como es el caso del lenguaje C con el que se desarrolló el S.O. UNIX o Linux, o el Basic del DOS.
- **Un intérprete de comandos**: El intérprete de comandos o mandatos es la interfase visible del S.O. con el usuario y es un módulo instalable, y hasta puede ser distinto para cada usuario.
- **Una biblioteca[4] de funciones**: Es la biblioteca en que se guardan las funciones ya programadas que utiliza el traductor (Compilador y Editor de Enlaces), para construir el Programa ejecutable.
- **Una biblioteca de comandos**: Los comandos se agrupan en una biblioteca y se cargan para su uso en Memoria Central (Por ejemplo, DOS).

[4] El Término Librería (Bookstore) es incorrectamente utilizado por Biblioteca (Library)

- **Un Software de comunicaciones**: El S.O. se ocupa de otras cosas además de las comunicaciones. Sus funciones son más amplias.

En general podemos decir que los S.O. cubren cuatro objetivos fundamentales:

a) facilitar el trabajo al usuario.
b) gestionar en forma eficiente los recursos.
c) inicializar la máquina.
d) Protege a los usuarios y a los recursos.

Entendemos por **Sistema Informático** (S.I.) = Hardware (HW) + Sistema Operativo (S.O.) + Programas de Aplicación + Usuarios + ambiente.

Un Sistema Operativo: es, entonces, una parte importante de cualquier sistema de computación.

Un sistema de computación o de procesamiento de datos: puede dividirse en cuatro componentes: el *hardware*, el *Sistema Operativo*, los *programas de aplicación* y los *usuarios*.

El hardware: (Unidad Central de Procesamiento + memoria central y Memoria secundaria + Módulo de entrada / salida + dispositivos de entrada / salida + comunicaciones), proporciona los recursos de computación básicos.

Los programas de aplicación: (compiladores, sistemas de bases de datos, vídeojuegos, programas comerciales, etc.), definen la forma en que estos recursos se emplean para resolver los problemas de computación de los usuarios.

En resumen: se podría decir que los Sistemas Operativos son un conjunto de programas que crean la interfase entre el hardware y el usuario, y que tiene cuatro funciones primordiales, que son:

- **Gestionar el hardware**: Se refiere al hecho de administrar de una forma más eficiente los recursos de la máquina.
- **Facilitar el trabajo al usuario**: Permite una comunicación con los dispositivos de la máquina.
- **Brindar un ambiente de ejecución seguro**. Protegiendo el uso correcto de los recursos.
- **Inicializar la máquina**.

El Sistema Operativo provee los medios para utilizar de forma correcta los recursos de un sistema de computación; no hace ninguna función útil por sí mismo. Simplemente provee un entorno en el cual otros programas pueden hacer un trabajo útil.

Podemos ver al SO como un **asignador de recursos**. Un sistema tiene diversos recursos que pueden requerirse para resolver un problema: tiempo de CPU, espacio de memoria, espacio de almacenamiento de archivos, procesos, dispositivos de E/S, etc. El SO actúa como gerenciador de esos recursos y se los otorga a programas específicos y usuarios a medida que son necesarios. Como puede haber diversos pedidos conflictivos para cada recurso, el SO tiene qué decidir que pedidos son atendidos y cuáles no. Su trabajo principal es mantener **un conjunto de tablas** en las cuales se describen los recursos y a que procesos están asignados los mismos.

Un SO es un programa de control que controla la ejecución de programas de usuario para prevenir errores y uso impropio de la computadora y es el responsable de la operación y control de dispositivos de E/S.

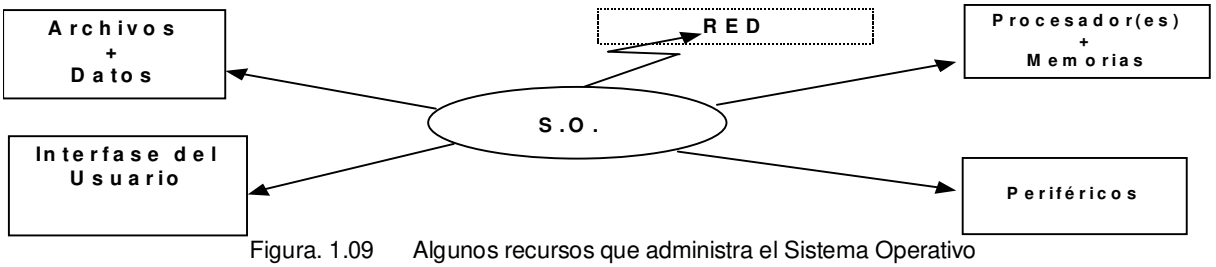


Figura. 1.09 Algunos recursos que administra el Sistema Operativo

1.1.2. Funciones de los sistemas operativos: a) Inicialización del sistema b) máquina extendida, c) El sistema operativo como administrador de recursos, d) Protección y seguridad.

1.1.2. Funciones de un sistema operativo.

Las funciones de los S.O. son básicamente cuatro: inicialización, máquina extendida, administración de recursos brindando servicios, y Protección y seguridad.

Veamos cada uno de estos puntos:

a) Inicialización del Sistema

Inicializar la máquina: Esta tarea es llevada a cabo por rutinas residentes en memoria ROM o RAM de la máquina más otras residentes en el disco del sistema o la partición activa para la función de arranque.

La inicialización tiene por objetivo preparar la máquina real y llevarla a un estado tal que pueda ejecutar el primer trabajo. Menos del 1 % de las tareas del S.O. se destinan a ésta función.

Hay dos tipos de Inicialización:

- **total** (cold *start)*, o *Bootstrappíng* o *Initial Program Loader (I.P.L.).*
- **parcial** *(warm start).*

1) Inicialización total (cold *start):*

El Sistema Operativo inicialmente se encuentra almacenado en la memoria secundaria. Al encender la máquina se carga y ejecuta un pedazo de código que se encuentra en la Memoria Central (ROM), el cual carga el BIOS (**B**asic **I**nput **O**utput **S**istem), y este a su vez carga el Nucleo o Kernel del Sistema Operativo en Memoria Central (RAM). Inicializa la máquina y carga todos los programas de aplicación y software necesario para permitir ejecutar programas o comandos del Shell.

Lo primero que se realiza es colocar la dirección 0 (o un valor predeterminado) de la memoria en el program counter (PC) del Procesador Central. Generalmente esta es una dirección ubicada en la memoria ROM, donde se encuentra la primera instrucción de la rutina de inicialización del hardware (BIOS). Luego de realizar todas las tareas de inicialización y control del hardware existente en el computador, la última instrucción del BIOS llama a la rutina de inicialización del sistema operativo (normalmente ubicada en el primer sector del disco del sistema- *bootsector).* A partir de este momento se carga el Kernel del sistema operativo, (quedando instalado permanentemente en memoria central, hasta que se ejecute el shutdown –cierre- del sistema) y por último carga el shell.

La rutina de inicialización del sistema operativo es la primera que se ejecuta. Dicha rutina es conocida como el **I.P.L.** ("Booteo" en la jerga informática), y una vez que dicha rutina es finalizada la máquina queda preparada para ejecutar la primer carga de trabajo bajo el control del S.O., que luego administrará todo el procesamiento del sistema de cómputo hasta su cierre. En UNIX dicha rutina es el init, y tiene 6 niveles distintos de inicialización.

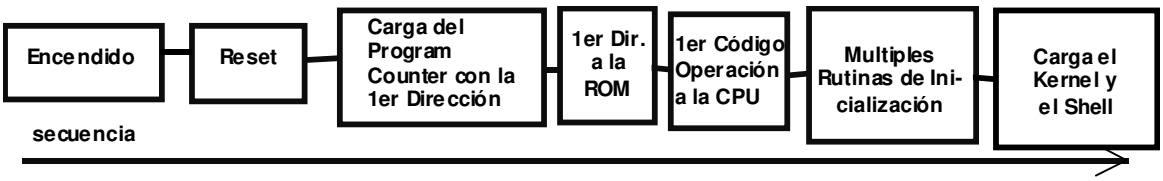


Fig. 1.10 Proceso de inicialización en frío

Existen dos tipos de verificación de recursos, la primera es realizada por el BIOS, y la segunda es

realizada por la rutina de inicialización del sistema operativo.

El propósito al verificación los recursos, es el de crear las tablas necesarias, para luego poder administrar los recursos disponibles en el sistema (Memoria, discos, impresoras, terminales, etc.).

Una vez terminada la rutina de inicialización, la máquina queda preparada para su uso presentando en la pantalla el símbolo (Prompt), respectivo, o solicitando el ingreso (login), al sistema. Todo el proceso de inicialización se realiza en modo monousuario.

2) Inicialización parcial (Warm Start):

En este caso no se ejecutan las rutinas de verificación de recursos (residentes en la BIOS o Kernel), sino que el proceso arranca directamente de la ejecución del proceso de inicialización del sistema operativo. La recuperación depende generalmente del tipo o clase de evento ocurrido que provocó la interrupción de la ejecución y de ello depende qué trabajo preservado reinicia. Los eventos causantes de la interrupción de la ejecución pueden ser generados por errores del Hardware, violaciones a Protecciones, corte de luz, finalización del tiempo de uso de CPU, etc. El ejemplo más común de éste tipo de inicialización es el CONTROL-ALT-DEL, existente en el DOS.

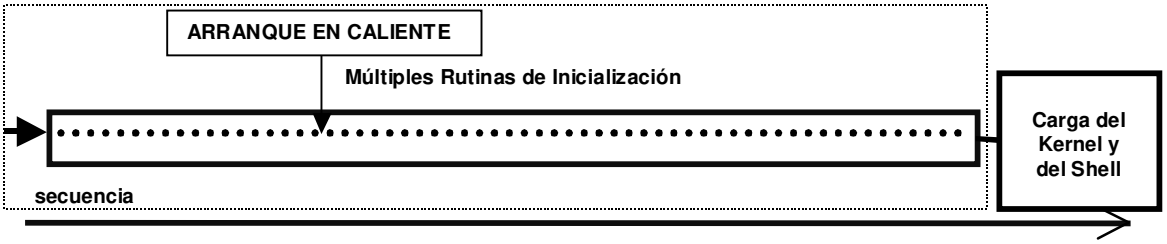


Fig. 1.11 Proceso de inicialización en caliente

El esquema para la inicialización parcial es importante en los casos de Sistemas en Tiempo Real, en grandes Sistemas Multiusuarios, o en Arquitecturas Tolerantes a fallas (fault-tolerance systems). En este último caso depende de la operatividad de los equipos.

b) Máquina extendida o interfase hombre - máquina.

El S.O. debe presentar al usuario el equivalente de una *máquina ampliada*, o *máquina extendida*, o *máquina virtual* que debe ser más fácil de usar o programar que el hardware implícito. Es decir, debe ocultar toda la complejidad de la máquina real para transformarse en una "máquina" más fácil de usar (más *"amigable"* o sea *User Friendly).*

Generalmente la visión de los usuarios de esta interfase se conoce con el nombre de SHELL. El Shell es la porción del S.O. que se encarga de hacer de interfase Hombre - Máquina y de transmitir al S.O. los requerimientos del usuario al sistema. La interfase humana es la que nos permite manipular el sistema y es la parte visible del S.O. y puede ser intercambiable.

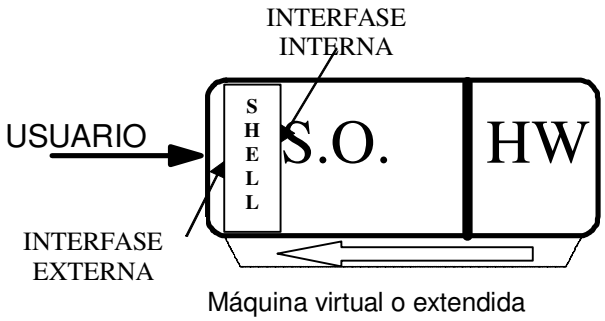


Fig. 1.12 Concepto de Máquina Extendida

Actúa como medio de comunicación entre el usuario y la máquina, y generalmente es realizada por un módulo del S.O. llamado **Shell** o **intérprete de comandos**. Tiene por objetivo servir de interfase Hombre-Máquina, o sea, actuar como una interfase interna (transparente al Usuario), y externa (de

comunicación con el Usuario). Entre el 7 al 9 % de las tareas del Sistema está dedicada a esta función.

Las funciones principales de la máquina extendida son:

- Separar la complejidad de la máquina desnuda al usuario. Una parte del S.O. se ocupa de eliminar la complejidad del Hardware y lo transforma en una máquina virtual más fácil de usar con una interfase para el usuario más amigable y menos compleja.
- Actuar como interfase de Entrada/Salida (E/S), y controlar el complejo manejo de los dispositivos de E/S cuando el usuario requiere de un servicio.
- Facilitar la comunicación con el usuario (mediante el SHELL), ya sea a través del Interprete de comandos, o del *Job Control Language* (J.C.L.), u órdenes verbales, o a través de Interfase Gráfica compuesta por iconos.
- Aceptar entradas de nuevos Trabajos.
- Proteger el sistema, permitiendo solamente el acceso a usuarios correctamente identificados y autorizar en el uso del sistema, a través de distintos métodos de autentificación (usuario / clave, reconocimiento de huellas dactilares, etc.)

El sistema operativo como máquina virtual: Un computador se compone de uno o más procesadores o CPUs, memoria central o RAM, memoria secundaria (discos), tarjetas de expansión (tarjetas de red, módems y otros), monitor, teclado, mouse y otros dispositivos. O sea, es un sistema complejo. Escribir programas que hagan uso correcto de todas estas componentes no es una tarea trivial. Peor aún si hablamos de uso óptimo. Si cada programador tuviera que preocuparse de, por ejemplo, como funciona el disco duro del computador, teniendo además siempre presentes todas las posibles cosas que podrían fallar, entonces a la fecha se habría escrito una cantidad bastante reducida de programas.

Es mucho más fácil decir *escribir "Hola" al final del archivo "datos"*, que poner en determinados registros del controlador de disco la dirección que se quiere escribir, el número de bytes que se desea escribir, la posición de memoria donde está la información a escribir, el *sentido* de la operación (lectura o escritura), amén de otros parámetros; y decir al controlador que efectué la operación; luego esperar; decidir qué hacer si el controlador se demora más de lo esperado; interpretar el resultado de la operación (una serie de bits); reintentar si algo anduviera mal, etc. Como se ve, esto es complejo, y además, habría que re-escribir el programa si se instala un disco diferente o se desea ejecutar el programa en otra máquina.

Todo esto indica que es necesario encontrar algún medio para aislar a los programadores de las complejidades del hardware. Esa es precisamente una de las tareas del sistema operativo, que puede verse como una capa de software que maneja todas las partes del sistema, y hace de *intermediario* entre el hardware y los programas del usuario. El sistema operativo presenta, de esta manera, una interfase o máquina virtual que es más fácil de entender y de programar que la máquina "binaria". Además, para una misma familia de máquinas, aunque tengan componentes diferentes (por ejemplo, monitores de distinta resolución o discos duros de diversos fabricantes), la máquina virtual puede ser idéntica: el programador ve exactamente la misma interfase, y por lo tanto los programas son los mismos.

El S.O. provee una máquina virtual, es decir, un ambiente en el cual el usuario pueda ejecutar programas de manera conveniente, protegiéndolo de los detalles y complejidades del hardware.

c) El sistema operativo como administrador de recursos

Más del 90% del SO se dedica a esta función. Las principales tareas consisten en asegurar que durante la ejecución se cumpla con los siguientes objetivos:

- Facilitar a los usuarios compartir y proteger los recursos.
- Optimizar el porcentaje de utilización de los recursos.

Para ello el S.O. controla todos los objetos de un sistema de cómputo complejo (usuarios, procesadores, memoria, discos, etc.), en cuanto a quién usa cuál recurso y su respectiva planificación del reparto. De esta forma va asignando y desasignando el uso de estos recursos e intercede como *arbitro imparcial* en los conflictos que se generan entre los programas y usuarios por el uso compartido.

El concepto de recurso computacional es una abstracción de los objetos, estructuras de datos, variables, rutinas, etc. que están accesibles a los programas o los usuarios a través del Sistema Operativo.

Cada uno de los administradores de los recursos, básicamente debe:

- Mantener actualizado un registro o contabilidad del estado o uso de los recursos.
- Cumplir (en función de la política implementada a tal fin), con las demandas, decidiendo quién, cómo, cuándo y por cuánto tiempo recibe determinado recurso.
- Asignar dicho recurso a quien lo demande y modificar su estado (de libre a ocupado).
- Recuperar el recurso después que se ha utilizado y actualizar su estado (de ocupado a libre).
- Función de Protección y tratamiento de errores.

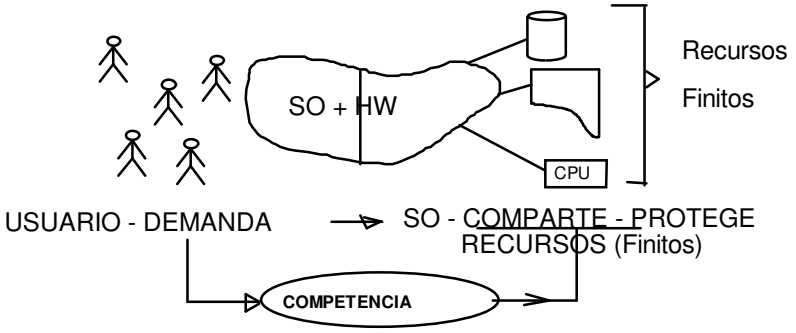


Fig. 1.13 El S.O. como administrador de recursos

Se dice que el *SO implementa* como *arbitro imparcial:*

- **Una Política**: dado que asigna prioridades (de uso y/o de acceso a los recursos).
- **Una Estrategia**: ya que ordena los accesos y los conflictos.
- **Una Autoridad**: pues debe recuperar los recursos otorgados a los procesos y ordenar el uso de los mismos.
- **Una Protección**: brindando seguridad a los usuarios entre sí y preservando la integridad de los recursos.
- **Una Contabilidad**: para llevar el control del uso y disponibilidad de los recursos.

y el *S.O. ofrece:*

- Facilidades para crear, manipular y eliminar objetos sobre los que se quiere realizar operaciones, a través de la Gestión y Conservación de la Información sobre ellos.
- Un ambiente para la ejecución de trabajos, mediante la gestión del conjunto de recursos que permiten ejecutar los mismos.
- Facilidades para compartir el conjunto de recursos entre los usuarios, mediante un planeamiento y ordenamiento de los trabajos.

La administración del sistema consiste en el:

- **Manejo y conservación de la información**: ofrece a los usuarios facilidades para crear, recuperar y eliminar objetos sobre los que se quieren realizar operaciones.
- **Manejo del conjunto de recursos**: que permiten ejecutar programas: El sistema crea el ambiente necesario para la ejecución de los trabajos.
- **Manejo y reparto del conjunto**: de recursos entre los usuarios mediante un planeamiento y ordenación de trabajos.

El sistema operativo administra eficientemente los recursos del computador sin generar conflictos. Por ejemplo: cuando hay dos o más programas que ejecutan simultáneamente y requieren usar el mismo recurso (como tiempo de CPU, memoria o impresora).

Además, en un sistema multiusuario, suele ser necesario o conveniente compartir información, además de dispositivos físicos,. Al mismo tiempo, debe tenerse en cuenta consideraciones de seguridad: por ejemplo, la información confidencial sólo debe ser accedida por usuarios autorizados. Un usuario cualquiera no debiera ser capaz de sobre escribir áreas críticas del sistema, etc. (En este caso, un usuario puede ser una persona, un programa, u otro computador).

En resumen, el sistema operativo debe llevar la cuenta acerca de quién está usando qué recursos; otorgar recursos a quienes los solicitan (siempre que el solicitante tenga derechos adecuados sobre el recurso); y arbitrar en caso de solicitudes conflictivas.

d). Protección y Seguridad

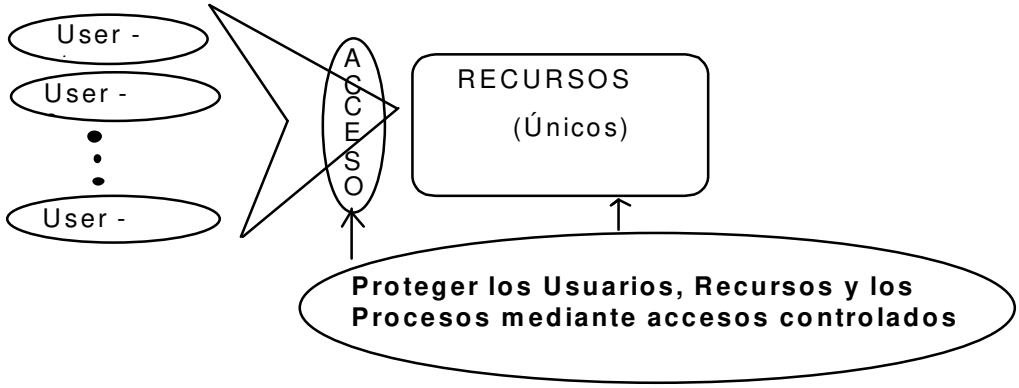


Fig. 1.14 Concepto de Protección

Objetivo: Garantizar la integridad de los recursos y de los Procesos, como también autenticar y validar a los usuarios en el sistema.

Al respecto se desarrollará un módulo completo sobre este tema, dado que el S.O. implementa distintas estrategias y mecanismos de protección para cada tipo de recurso.

Para resolver este conflicto el S.O. ofrece los siguientes mecanismos:

1. **Ejecución dual de instrucciones**: *Maestro - Esclavo* (este concepto se amplía en un próximo punto de éste módulo).
2. **Mutua Exclusión**: (consiste en asegurar que los recursos compartidos sean accedidos por ***un solo*** proceso a la vez bajo ciertas condiciones (lo veremos más detallado en el módulo 4).
3. **Control de accesos.**

1.2. Evolución histórica de los Sistemas Operativos. a) Sistema de procesamiento “Batch”. b) Sistema "Spooling". c) Sistemas Interactivos. d) Sistemas Distribuidos. e) Tendencias.

1.2. Evolución histórica de los Sistemas Operativos.

No solo hubo una evolución en el hardware y en el software de los equipos, sino también en el terreno de los S.O..
Dos son las consecuencias de ésta evolución:

- Cada vez son mayores las posibilidades de los sistemas operativos: pueden controlar a un mayor número de periféricos asociados al computador y ponen a disposición del usuario un repertorio de comandos mas potentes y además permiten poner en práctica nuevos métodos mas eficaces y rápidos para explotar las posibilidades del hardware de la máquina.
- La llegada del microprocesador y su implantación como corazón de las computadoras permitió un gran avance en la estandarización de los S.O., destacándose entre ellos, el DOS, el OS-2, VM, el UNIX, Linux, Solares, etc.

Los Sistemas Operativos fueron desarrollados como respuesta a la necesidad de aumentar la utilización de los procesadores y de los dispositivos periféricos, y esto sigue siendo una de las funciones primordiales de los Sistemas Operativos, sobre todo, el mejorar el uso del hardware al automatizar la corriente de trabajos y realizar decisiones sobre el manejo de los recursos del sistema en la escala de tiempos de la computadora y no en la del operador humano.

En los primeros días de la computación, la computadora era exclusivamente operada a mano. En éstas condiciones, los usuarios tenían toda la máquina para sí por algún intervalo de tiempo. Entonces el usuario disponía de todos los recursos y cualquier administración de esos recursos estaba dedicada a hacer mas conveniente la operación de la máquina por parte del usuario. Esta forma de usar la máquina se dió en llamar “*Open Shop*” que era monoprogramado y monousuario.

Cuando la comunidad de usuarios creció, se hizo necesario asegurar una distribución eficiente de los recursos físicos del sistema entre todos.
A partir de 1950, surgen nuevas técnicas de administración y ordenamiento con el propósito de aprovechar los equipos de la mejor manera posible. Estas técnicas emergen debido a que varios usuarios comparten una instalación y generan colas de trabajos a ser ejecutados, por lo que deben adoptarse decisiones sobre el orden en que serán atendidos, entonces se establecen ciertas reglas llamadas "***Scheduling Algorithms" o algoritmos de administración o planificación.***
Los primeros S.O. surgen a partir de 1956 al reemplazar las funciones que realizaba el usuario. El S.O., en su evolución, atraviesa distintas etapas hasta llegar a su estado actual.
La primera técnica fue el "***Open Shop"*** (1956-1960), y en ella el usuario disponía de un intervalo de tiempo, por ejemplo 15 minutos, durante el cual toda la máquina estaba a su disposición para operarla y usarla.
Este método de administración, entregaba el sistema de cómputo a un solo usuario por vez lo cual significa que cada usuario usaba la máquina un intervalo de tiempo, durante el cual podía ejecutar sus programas, localizar errores, experimentar, etc..
Antes de trabajar sobre la máquina, cada usuario debía reservar un intervalo de tiempo inscribiéndose en una lista, lo que le permitía durante ese tiempo tener a su disposición la computadora para operarla y usarla en forma exclusiva.
Sin embargo, esa forma de operar, hacia que la computadora estuviera la mayor parte del tiempo ociosa, debido al tiempo que perdían los usuarios haciendo un uso indiscriminado de la misma.
Aparece entonces lo que se dió en llamar "***Closed Shop"***. El Closed Shop es una forma de trabajar que si bien es secuencial, los usuarios debían entregar sus trabajos (job) a un operador, quien pasa a tener un papel preponderante, dado que era el "dueño, amo y señor" de un recurso complejo y caro y disponía del orden para llevar a ejecución los trabajos. Estos abusos motivaron a resolver el problema mediante una programación (software) y así nace un incipiente S.O. que reglamentaba el uso del computador.
Es así como surgen nuevos Sistemas Operativos, que impiden al operador interactuar libremente con la máquina. Estos S.O. ponen en práctica un procedimiento llamado "*Closed Shop"*, mediante el cual el usuario dejó de disponer libremente de la computadora. Este tipo de procedimiento fue posible gracias a esos nuevos S.O. que permitían realizar, cada vez mas, un mayor número de tareas.
Esto produjo una evolución importante ya que el usuario entregaba al operador un lote de tarjetas listo para procesar. Éste armaba una cola de tarjetas (según su orden de llegada), y el procesador las leía cargándolas en memoria central y luego las ejecutaba devolviendo los resultados impresos. Esto se esquematiza en la Fig. 1.15.
Luego de procesar todos los trabajos, y recién cuando el último trabajo había sido concluido, los resultados eran listados, de a uno por vez, por la impresora.
La lentitud de la Entrada / Salida (E/S) en estos sistemas constituía un "cuello de botella" debido al tiempo empleado para la E/S. Además esa forma de trabajar era muy onerosa como consecuencia de los intervalos de procesador ocioso por la pérdida del tiempo en que los usuarios hacían un uso indiscriminado del computador.
Es así como se buscan nuevas técnicas que permitan mejorar en la práctica el procedimiento llamado "Closed Shop" (1960-1963) mediante el cual el usuario dejó de disponer libremente de la máquina y debía entregar sus trabajos al Operador del Centro de cómputos.
Esta automatización en la administración provocó algunos inconvenientes, tales como: ¿Cómo proteger el S.O. residente contra los programas cargados erróneamente sobre él?, o ¿Cómo obligar a los usuarios a devolver el control al S.O. una vez que terminaban ?, etc..
Durante este período, el problema más serio lo constituían la diferencia de velocidades que existía entre los dispositivos de E/S y el procesador central.
Para corregir este desequilibrio, aparecieron dos mejoras. Ambas de gran importancia para los S.O..
La primera fue la introducción del ***canal de E/S.*** Esta mejora del hardware fue capaz de controlar uno o varios dispositivos periféricos, y que una vez puesta en marcha funcionaba independientemente del procesador central, permitiendo que las entradas o salidas se ejecutaran superpuestas con la ejecución de la CPU.
La segunda fue la introducción del procedimiento conocido como "***Entrada/Salida fuera de línea" (Off Line***). En vez de que la computadora usara a los lentos dispositivos periféricos en forma directa, la entrada era transcripta de tarjetas a cintas magnéticas y un programa residente en memoria central recibía la entrada leyendo de las cintas las "imágenes" de las tarjetas. Este programa pasó a formar parte del S.O. llamándose **cargador** (**loader)**.
Una vez establecido el principio de E/S fuera de línea sobre cinta, quedó abierto el camino para el desarrollo de un sistema que encadenara la secuencia de trabajos. A éste sistema, que ejecutaba en

serie y en forma automática las tareas contenidas en una cinta magnética, se lo llamó "***Sistema de procesamiento por Lotes (batch)"***.

En este sistema no es precisa la atención del usuario durante la ejecución de los trabajos. Una vez que un Trabajo o JOB se presentaba para su tratamiento, el usuario perdía la posibilidad de intervenir y modificar su desarrollo.

Probablemente el primer sistema de este tipo haya sido el S.O. de la organización de usuarios SHARE (Share Operating System), antepasado del "Sistema Monitor FORTRAN" (F.M.S.) que fuera ampliamente y vastamente utilizado por esa época.

Otro hecho notable en la evolución de los sistemas operativos fue el advenimiento del "***tiempo compartido" (Time Sharing***), el cual tuvo su origen en una evolución del hardware que introdujo el **concepto de interrupción**, el cual permitió sincronizar el procesador central con los dispositivos periféricos al completarse la E/S.

De ahí se pasó rápidamente a la ***multiprogramación***. La primera computadora que introdujo la idea de tener un sistema operativo complejo para controlar varios programas en ejecución simultánea fue la FERRANTI ORION, a la que siguió la FERRANTI ATLAS (luego ICL).

Entre los años 1967-1968 nace el concepto de ***"tiempo compartido"*** tomado como sinónimo de acceso múltiple (multiaccess), es decir, que si en una máquina hay varios programas simultáneos, el procesador es compartido por ellos en el tiempo y cada programa trabaja por turno durante un cierto lapso pequeño de tiempo, pareciendo que todas las tareas progresan simultáneamente.

Esto es atrayente si cada programa esta conectado al usuario por medio de una pantalla en línea, o alguna terminal semejante, ya que cada usuario tendrá la impresión de que dispone de toda la máquina para su uso exclusivo.

En síntesis, el "tiempo compartido", permitió, por intermedio del desarrollo del S.O., administrar una cantidad de usuarios que explotan y comparten los mismo recursos computacionales (es decir "hardware" y "software").

Luego, con la construcción de memorias de gran capacidad y acceso directo y la multiprogramación, se diseñaron S.O. capaces de manejar una corriente contínua de trabajo sobre una misma computadora.

Entre los años 1968-1969, se consolida el teleprocesamiento como técnica de explotación. Se entiende por ***teleprocesamiento*** a la técnica de empleo de las computadoras a distancia (desde terminales conectadas a través de líneas telefónicas, telegráficas, enlaces de radio, o algún otro medio de comunicaciones).

Mas tarde, toma incremento la utilización de la técnica de "***paginación***" (almacenamiento dividido en un número de "páginas" de tamaño fijo y constante que se utilizan para ejecutar y transferir datos y programas).

Esta técnica suministra un almacenamiento virtual mayor que la memoria real. De aquí surge la definición de ***memoria virtual*** como el espacio de direccionamiento de un usuario que no necesita tener todo el programa residente en Memoria Central para su ejecución con la condición de que dicho programa esté soportado en un ***Dispositivo de Almacenamiento Secundario de Acceso Directo (Direct Access Storage Device DASD***) como ser un disco. Entonces el Usuario no tiene que preocuparse por las limitaciones debidas a que su programa entre o no en la memoria central, ya que este tipo de contingencias es resuelto por el sistema de paginación provisto por el S.O..

Esta técnica de memoria virtual trae aparejado un notable avance ya que no limita el espacio de memoria central para cada usuario sino suministra un almacenamiento virtual que está parcialmente en el disco o en algún otro tipo de memoria masiva, y en parte en la memoria central, y usar la paginación para organizar la ejecución de la aplicación en el almacenamiento central en el momento de ejecución.

Esta técnica no hubiera sido posible sin la evolución tecnológica de los dispositivos de almacenamiento de acceso directo y del sistema de interrupciones.

En 1972-1973, aparece el concepto de ***firmware*** que implica disponer de un Software en un Hardware funcionando en un ambiente determinado. A partir de 1982 en adelante se busca la formalización y el refinamiento de los sistemas operativos que acompañan a cada microprocesador y su arquitectura desde los 8 bits a los 64 y 128 actuales.

La reducción de los costos ha favorecido los avances hasta llegar a compartir recursos en sistemas distribuidos geográficamente entrelazando a nivel mundial mediante las comunicaciones a redes complejas como lo es Internet.

A continuación detallaremos los principales sistemas de procesamiento que actualmente están en uso.

a) SISTEMA DE PROCESAMIENTO "BATCH".

Los primeros sistemas operativos, aunque resulten muy primitivos al ser comparados con los actuales, suponían una mejora al liberar al operador de tareas rutinarias. Desde ese momento se comenzó a buscar un método de administración del procesador que mantenga activo al mismo la mayor parte del tiempo posible.
La actividad de un procesador se puede determinar mediante las siguientes fórmulas:

$$\text{UP(utilización del Procesador)} = \frac{\text{tiempo de ejecución}}{\text{tiempo total}}$$

$$\text{P (producción)} = \text{número de trabajos ejecutados por unidad de tiempo}$$

Cabe señalar que una de las características de las máquinas de esa época era que solo admitían un trabajo por vez, y por lo tanto no podían, bajo ningún punto de vista, superponerse en el tiempo una operación de E/S con la ejecución de un programa.
Es así como se llega a un SO del tipo "batch", donde los usuarios entregaban al operador los **lotes** de tarjetas, con la diferencia que estas tarjetas se grababan en cintas magnéticas según su orden de llegada. Para realizar esta tarea, generalmente, se usaba una pequeña computadora externa al sistema, dedicada y de bajo costo, cuya única función era leer tarjetas y grabarlas en una Cinta. Así nació el concepto de dispositivo ***periférico.***
El esquema de un batch sería:

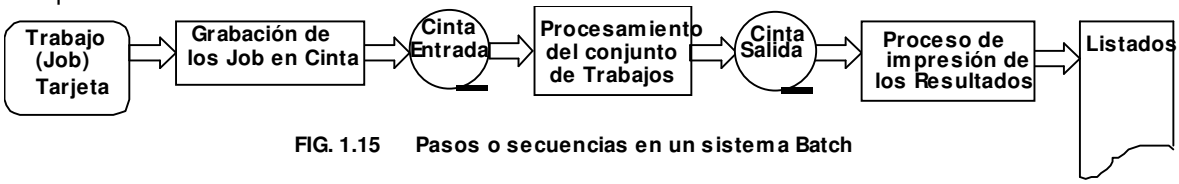


FIG. 1.15 Pasos o secuencias en un sistema Batch

Después de la grabación, la cinta era íntegramente leída por la computadora principal, para luego ser procesados los trabajos de a uno por vez, y una vez que el trabajo era terminado, todos sus resultados se grababan en una cinta, la cual era leída por otra computadora pequeña especializada llamada impresora de línea que generaba los listados de los resultados.
El usar un computador pequeña, además de la principal, permitía que durante la ejecución de un batch en la computadora principal, el operador pudiera utilizar la computadora pequeña para imprimir la salida de un batch anterior y ejecutar la grabación de entrada del batch siguiente.
De esta manera tanto la computadora principal como los dispositivos de E/S estaban ocupados el máximo tiempo lo que hacía posible la disminución de las demoras de la computadora principal al no realizar las E/S lenta y estas trabajar en paralelo superponiéndose en el tiempo las actividades de entrada y las de salida.
Colocando un solo trabajo por cinta se lograba un bajísimo rendimiento del procesador (menor a 10 %). Esto se debe a que el montaje y desmontaje se realizaba por el operador. Entonces se decidió colocar más trabajos por cada cinta. Esto mejoró el rendimiento llevándolo al 30 %. Para calcular el rendimiento se utilizaban las siguientes fórmulas:

$$\text{UP} = \frac{\text{TEB (tiempo de ejecución del batch)}}{\text{TRB (tiempo de respuesta del batch)}}$$

donde:

$$\text{TRB (tiempo de respuesta del batch)} = \text{tiempo de montaje del batch} + \text{tiempo de ejecución del Batch}$$

también se podría expresar:

$$\text{TRB (tiempo de respuesta del batch)} = \frac{\text{tiempo de montaje del batch}}{\text{1 - (Utilización del Procesador)}}$$

El objetivo era lograr el máximo de rendimiento del procesador. Hoy día aun se utiliza el concepto de batch para el procesamiento secuencial de ciertas aplicaciones.

b) SISTEMA "SPOOLING"

Para una operación exitosa de un sistema de multiprogramación deben tenerse en cuenta dos factores:
1) Diversidad de trabajos.
2) Disponibilidad de suficientes periféricos.

El primer factor, la diversidad de trabajos, es necesario para poder superponer las E/S adecuadamente con el uso de la CPU, y debe haber una cantidad suficiente de periféricos para que los distintos programas puedan operar simultáneamente. Por ejemplo, si se dispone de una sola impresora, y todos los programas requieren de salida impresa, les será imposible a los mismos acceder al dispositivo simultáneamente.

Un programa desarrollado en Inglaterra por la firma ICL, trata de resolver este problema, de modo tal que cada programa no solicite la impresora hasta que tenga algo para imprimir. Esta solución no fue satisfactoria, ya que las salidas de los distintos programas aparecían mezcladas en la impresión.

Surgió así una forma muy popular de operar, que se dió en llamar ***"Operación simultánea de periféricos en Línea" en inglés "Simultaneus Peripherical Operation On Line" o Spooling.***

Este sistema de operación simultánea de periféricos constaba de cuatro módulos o unidades:
1) El ***programa ejecutivo*** que organizaba la multiprogramación de los demás componentes.
2) El ***módulo de entrada*** que leía los trabajos de la lectora de tarjetas y los grababa en el disco.
3) El ***módulo de salida***, que leía los trabajos del disco y los depositaba en la impresora de línea.
4) El ***modulo de secuencia de trabajo***, que llamaba a los computadores, programas, etc..

Otra significativa innovación de ésta época fue la utilización de grandes memorias secundarias, lo que facilitó el desarrollo de los sistemas de spooling.

Un sistema de spooling puede esquematizarse como se observa en la Fig. 1.16.

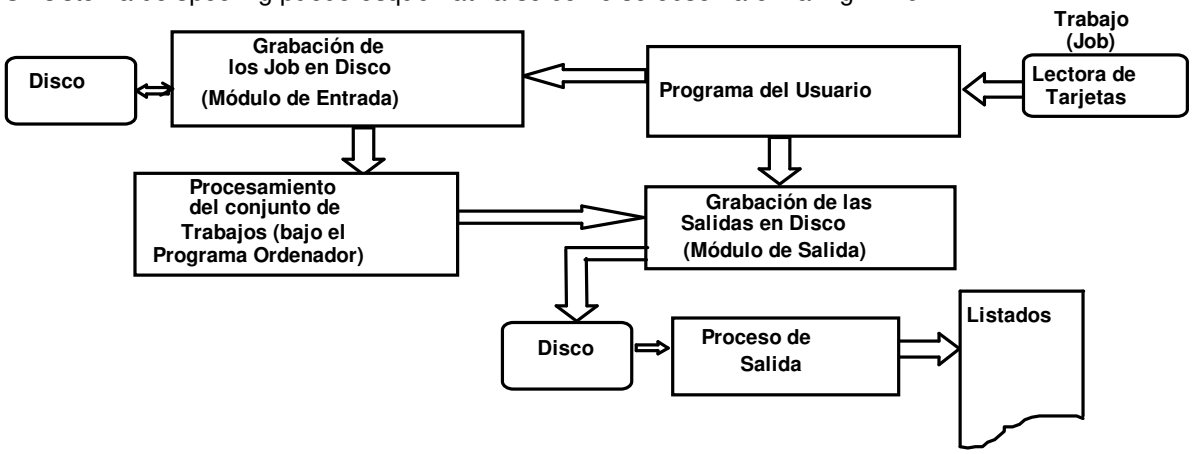


Figura. 1.16 Módulos en el procesamiento SPOOLING

Hoy día se utiliza este concepto en un sistema operativo como un módulo llamado spooler que se ocupa de administrar todas las impresiones y en algunos casos la captura de datos de entrada.

c) SISTEMAS INTERACTIVOS.

En el ***sistema interactivo***, el computador es capaz de generar una respuesta inmediata ante una ación o solicitud externa. Por consiguiente, la relación entre el usuario y la máquina es interactiva. Es multiprogramado.

Al introducir una orden, como por ejemplo a través del teclado, la computadora la ejecuta entregando el resultado inmediatamente. Cuando el usuario necesita mantener un diálogo constante con la máquina, por ejemplo, para recibir respuestas a requerimientos o consultas que se efectúan al computador, se debe optar por este tipo de sistema. La expresión "inmediato" debe entenderse que la computadora satisface los requerimientos de los usuarios a la velocidad del ser humano que es lenta comparada con la velocidad que procesa la computadora.

Una misma computadora puede responder a varios usuarios en pocos segundos cuando el tiempo de procesamiento de cada solicitud es muy corto y el diálogo se limita por el tiempo que requiere el usuario para pensar o actuar.
El tiempo de procesador que se requiere para atender cada transacción desde una terminal es del orden de los milisegundos y su organización no puede depender de estimaciones previstas por el usuario.
Esto obliga al administrador de Trabajos del S.O. a entregar el procesador durante breves intervalos de tiempo, de longitud fija a un grupo de usuarios que están usando el equipo, Esta regla llamada "***Round Robin***" asigna el tiempo de procesador en rebanadas. Estas porciones de tiempo reciben el nombre de ***"Time Slice***" o "**Quantum**" . A cada trabajo se le asigna un time slice fijo, y, si no se completa durante ese intervalo, el trabajo es interrumpido y retorna a la cola de trabajos en espera del procesador. Los trabajos que van llegando también se incorporan a la cola. Este tipo de sistema puede trabajar tanto en tiempo real como en tiempo diferido.
Trabajar en ***tiempo real*** significa que el computador responde a la velocidad en que se produce el fenómeno (Velocidad de respuesta en el tiempo que sea necesario, generalmente es a la velocidad del ser humano o en su escala de tiempos).

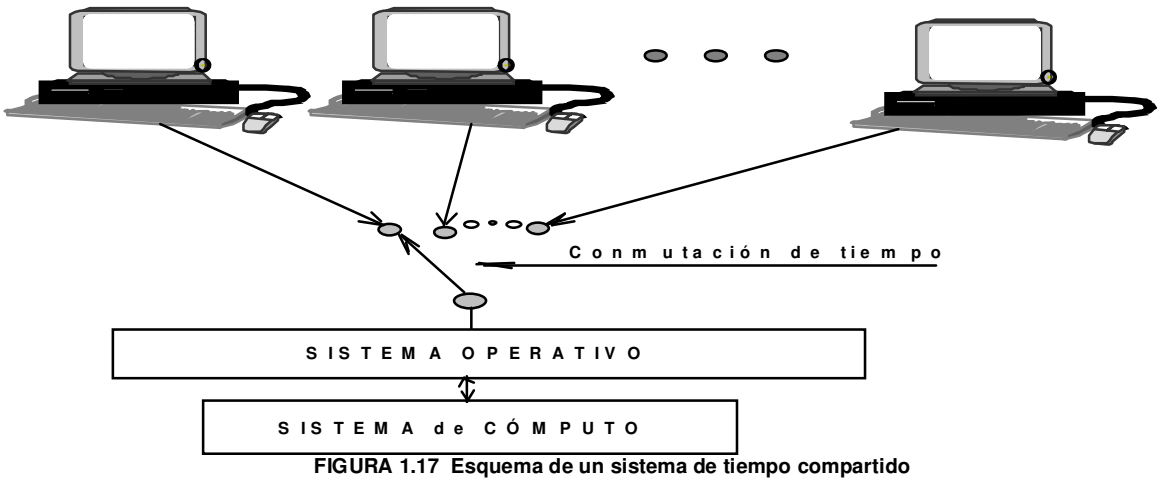


FIGURA 1.17 Esquema de un sistema de tiempo compartido

Trabajar en ***tiempo diferido*** significa que el usuario envía su programa o datos en forma remota y a partir de ese momento el programa queda fuera del control de usuario, y se ejecutará en algún instante posterior.
Si el programa ingresa en tiempo diferido, el S.O. lo identifica y procede a organizar su lista de programas a ejecutar, en el cual se agrega la prioridad que le corresponde entre todos los programas que también trabajan en tiempo diferido.
Si la computadora trabaja en tiempo real y hay recursos disponibles para ejecutar otros trabajos y en la cola hay jobs diferidos, el S.O. puede proceder a ejecutar el trabajo más prioritario de los diferidos simultáneamente con los de tiempo real dado que hay disponibilidad de recursos.
Si en el momento en que se ejecuta un trabajo en tiempo diferido, se presenta un trabajo en tiempo real, el que se estaba ejecutando es detenido para que se comience a ejecutar el de tiempo real. En éste caso no había demanda de ejecución en tiempo real. Este último siempre tiene máxima prioridad.
Si no hay recursos, los trabajos van engrosando la cola de programas a la espera de su ejecución.

d) SISTEMAS DISTRIBUIDOS

Se dio en llamar procesamiento centralizado al sistema de procesamiento de datos en que todas las funciones de procesamiento están centralizados en una CPU y un S.O. y en cambio, procesamiento distribuido al Sistema de procesamiento descentralizado de datos que se ejecutan en nodos dispersos geográficamente interconectados mediante una red y que utiliza una función de comunicaciones para la transferencia de datos o programas.
El procesamiento distribuido se caracteriza por realizar un procesamiento cooperativo entre distintas máquinas y en particular está caracterizado por:

1. Fragmentar los elementos que componen una aplicación sobre dos o más sistemas interconectados, de igual o diferente arquitectura operativa.

2. Los recursos de los sistemas cooperantes se controlan y administran en forma independiente
3. La relación entre ambos sistemas puede tomar diferentes formas. Por ejemplo:

- Client - Server es un modelo de procesamiento distribuido caracterizado en que uno de los sistemas cooperante asume el rol de cliente para solicitar un servicio específico a un proceso servidor.
- Peer to Peer es otro modelo en que cada nodo es igual a otro (cada uno es un par del otro). Cada uno ofrece los mismos servicios.

El objetivo básico es buscar un procesamiento rápido en paralelo y de este modo incrementar la eficiencia para lograr un mayor rendimiento general del sistema.

e) TENDENCIAS.

De lo dicho podemos extraer de la historia retrospectiva, que inicialmente el hardware era sencillo, aislado, simple, cada individuo podía emplear en forma exclusiva al Sistema y como se fueron presentando algunos problemas de organización, se produjo un gran avance tecnológico desde ese estado al de los grandes Sistemas, con complejas organizaciones de multiprogramación, multiprocesadores, sistemas de memoria virtual, etc. para luego, con el advenimiento de los chips y los microprocesadores pasar a sistemas económicos actuales como se las Computadoras Personales (PC) y estaciones de trabajos conectadas en redes como servidores.

Las tendencias se pueden resumir en:

- de un usuario, a varios usuarios sobre un mismo Sistema o sistemas distribuidos.
- de usuarios aislados, a usuarios que comparten y cooperan entre sí.
- de una operación secuencial en Batch a la multiprogramación y al tiempo compartido.
- De un procesamiento centralizado a un procesamiento distribuido basado en las comunicaciones.
- De un sistema de procesamiento en redes locales a grandes redes del tipo Grids
- De un servidor a un conjunto de servidores en paralelo (Clusters)
- tanto en el hardware como en el software, presentan un aumento en el paralelismo y concurrencia de ejecución.
- y, muy importante, una tendencia de la administración de recursos simples a complejos no físicos.

1.3 Características necesarias en el HARDWARE:

1.3.1 Elementos de protección:

a) Necesidad de protección en el sistema operativo.

b) Juego de Instrucciones diferenciadas del Procesador.

c) Modo dual de ejecución del procesador

1.3.1. Elementos de protección:

a) Necesidad de protección en el Sistema Operativo.

El sistema operativo debe ofrecer mecanismos que permitan proteger el hardware, restringiendo tanto a usuarios como a aplicaciones de realizar ciertas operaciones que podrían afectar la estabilidad del sistema, la integridad y la disponibilidad de la información.

Los mecanismos de protección del hardware que ofrece el sistema operativo no pueden asegurar la integridad física de los dispositivos, es decir, no se puede evitar que se dañen los equipos por intervención del ser humano pero sí puede evitarse que se dañe la información contenida en los sistemas por uso incorrecto de los mismos.

Con esto podemos concluir que, si bien no se ofrece un mecanismo de protección física que asegure la integridad de los equipos utilizados, el sistema operativo ofrece mecanismos de protección que aseguran el cumplimiento de los siguientes principios básicos de la seguridad:

- § Integridad
- § Disponibilidad
- § Acceso controlado

Es por esto que puede decirse que el sistema operativo es la base de la seguridad de cualquier sistema aplicativo. Tanto es así, que, según principios básicos de la revisión y auditoria de sistemas, una incorrecta configuración de los accesos y permisos en el sistema operativo implican que no se podrá confiar en la información almacenada en los sistemas, salvo que se puedan realizar pruebas que demuestren que la aplicación brinda un adecuado nivel de protección.

Es aún más importante el soporte que brinda el sistema operativo en funciones específicas como el acceso a recursos. Entre las funciones básicas de todos los sistemas operativos actuales se encuentra la autenticación, que permite asegurar que un usuario tiene ciertas autorizaciones y que es quién dice ser.

En resumen, la primera barrera de seguridad para el acceso a la información almacenada en un computador es el sistema operativo, que brinda mecanismos básicos de protección del hardware y de la información almacenada.

El fabricante del procesador también prevé elementos de seguridad como ser un juego de instrucciones diferenciadas y un modo de ejecución para que el S.O. pueda cumplir con la función de protección.

b) Juego de Instrucciones diferenciadas del Procesador

Los fabricantes de procesadores deben incorporar en sus diseños, una interfase basada en un juego de instrucciones *(instruction set),* las cuales permitirán ejecutar cualquier software sobre ese procesador. Este repertorio de instrucciones se dividirán en dos tipos: **instrucciones privilegiadas** (también llamadas Kernel), que el fabricante destina para la ejecución del Sistema Operativo y que no admiten ser interrumpidas durante su ejecución y otro grupo destinadas a los programas **comunes o del usuario**, que admiten ser interrumpidas en su secuencia de ejecución. Este conjunto de instrucciones se diseñan especialmente para que el hardware de ejecución brinde una alta eficiencia en el procesamiento

Las instrucciones privilegiadas o reservadas son sólo utilizables por el sistema operativo, y se definen generalmente en base a como están definidos los dos modos de funcionamiento del procesador antes mencionados.

El funcionamiento asincrónico de los canales de E/S; y la CPU fue una de las primeras formas de procesamiento múltiple, es decir, el funcionamiento simultáneo de dos o más procesadores vinculados con una misma memoria.

Muchas veces, el conjunto de instrucciones de E/S (utilizaremos las siglas E/S como sinónimo de Entrada-Salida), son completamente independiente de las ejecutadas por la CPU, y pueden ser realizadas asincrónicamente (simultáneamente), mientras la CPU se ocupa de otra tarea. Para ello cualquier procesador (Microprocesador o CPU), trae en su juego de instrucciones (*instruction set*), los dos tipos antes mencionadas: instrucciones privilegiadas reservadas para el Sistema Operativo e ***instrucciones comunes u ordinarias*** destinadas para el usuario.

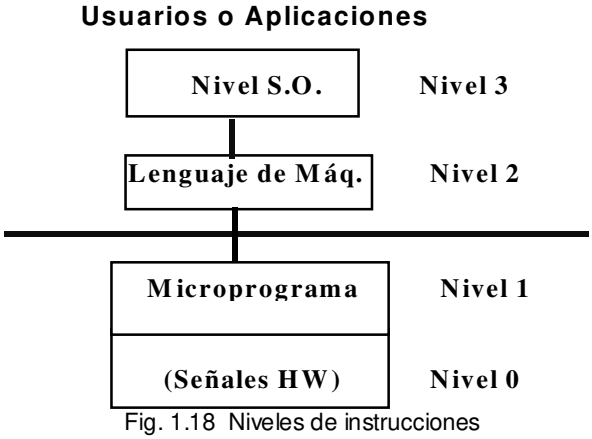


Fig. 1.18 Niveles de instrucciones

Privilegiado, **supervisor, monitor** o **Kernel** (modo de trabajo del S.O.)

- El S.O. ejecuta sin interrupción.
- Acceso a todos los recursos (toda el área de memoria es direccionable en este modo sin

restricciones)

En el Nivel 3 coexisten los dos tipos de Instrucciones:

1. **Privilegiadas o del S.O. (Supervisor, monitor o Kernel)**: Modifican el Estado del Procesador, operan los Procesadores de E/S, Modifican la Protección del área de Memoria Central, etc.

Ocurren en los siguientes casos:

1) Cambio de Modo de ejecución del procesador: Usuario a Kernel
2) Llamadas al sistema para un servicio requerido por el programa que se está ejecutando (System Call).
3) Cuando ocurre una Interrupción por algún evento en el sistema.
4) Cuando ejecuta el S.O.
5) Cuando se produce un error en la ejecución.

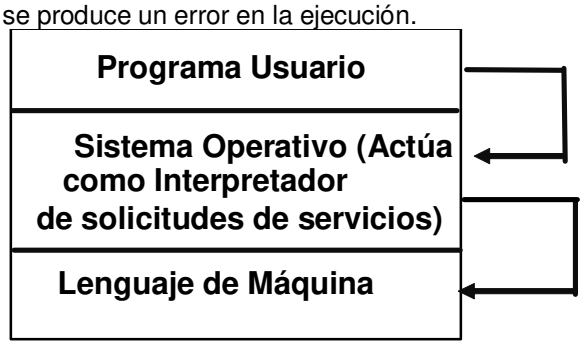


Fig. 1.19 Instrucciones con intervención del S.O. (SYSCALL)

2. **Ordinarias o del Usuario** (Comunes): No puede ejecutar las Instrucciones Privilegiadas (lo intercepta el S.O.), y solo ejecuta la secuencia de instrucciones del usuario. Las características de este modo de funcionamiento son:
   1) Ejecuta un programa usuario.
   2) Puede ser interrumpido.
   3) Acceso restringido a recursos e instrucciones del procesador.
   4) Ejemplo: Instrucciones Aritméticas, Lógicas, de Corrimientos, etc.

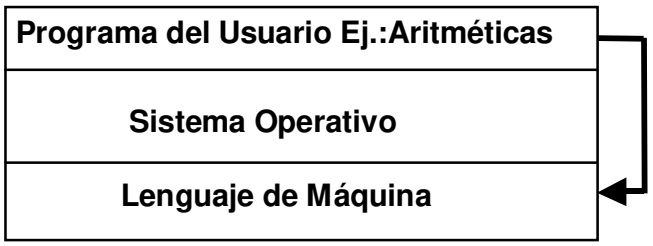


Fig. 1.20 Instrucciones sin intervención del S.O. (Usuario)

c) Modo dual de ejecución del procesador.

La segunda característica que debe incorporarse al procesador es su modo de funcionamiento, que está íntimamente relacionado con el repertorio de instrucciones.

El modo de funcionamiento dual o sea en dos estados. Uno privilegiado o modo Kernel, en que ejecuta instrucciones del S.O. y el otro modo usuario en que ejecuta instrucciones comunes.

El Hardware detecta muchos de los errores ocasionados por los programas que están ejecutándose y normalmente son tratados por el S.O. como un servicio de interrupciones, pero no todos los errores están previstos, además depende de la modalidad del procesamiento: si es por lotes (Batch), o interactivo.

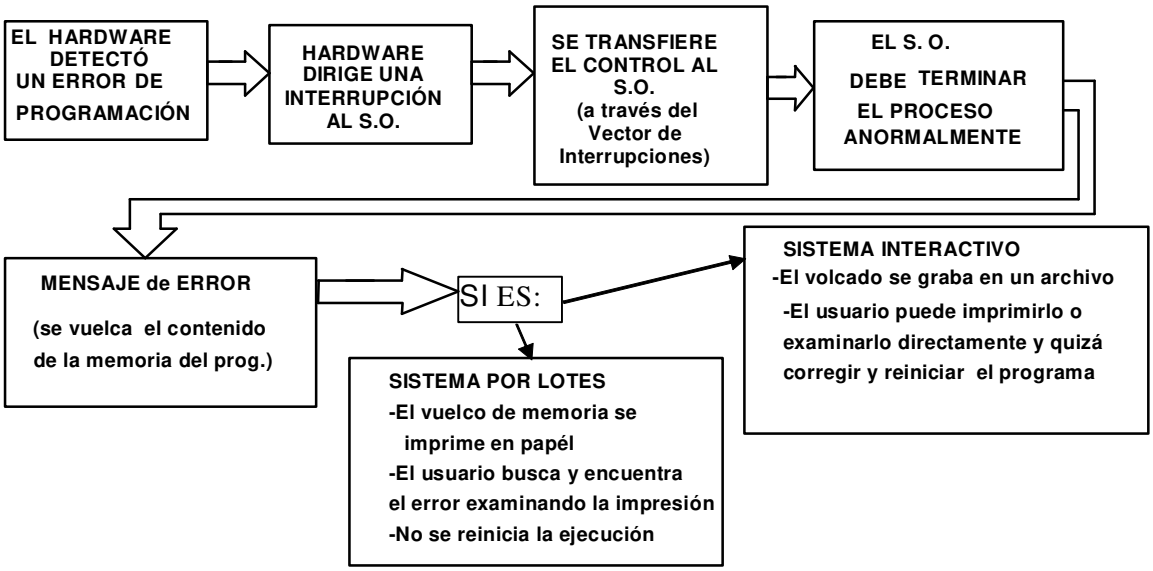


Fig. 1.21 Secuencia de un error tratado por el S.O. de un mainframe

De esta forma se logró proteger al S.O. de violaciones cometidas por los usuarios y también se protegió a los usuarios entre ellos mismos, simplemente clasificando como instrucciones privilegiadas, aquellos que solo pueden ejecutarse en modo Kernel, como ser las instrucciones que pueden causar algún daño si son mal utilizadas por los programas de los usuarios (Por ejemplo: un direccionamiento a áreas de memoria central no autorizadas).

Decíamos que este sistema es insuficiente para asegurar que:

-se detecten todos los errores
-proteger al S.O.
-proteger a los demás programas y datos;
-proteger cualquier recurso compartido.

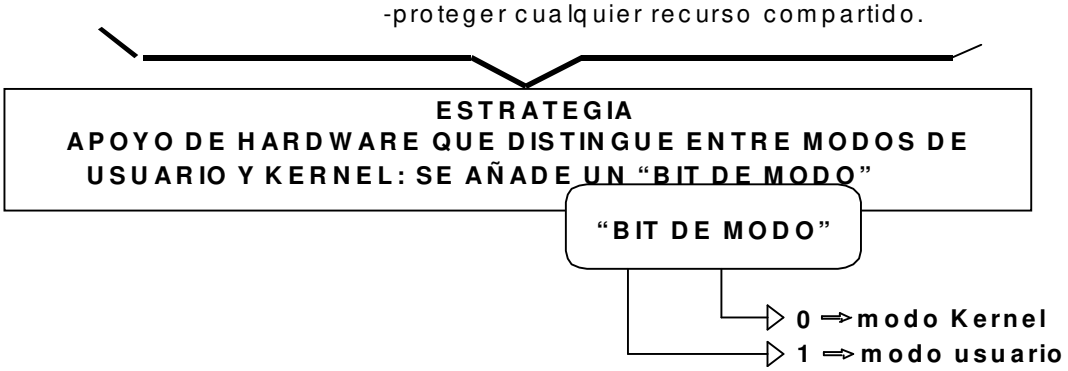


Fig. 1.22. Estrategias en el modo de ejecución del hardware

Debido a éstos problemas se diseñaron instrucciones privilegiadas para los siguientes casos:

- instrucciones de Entrada - Salida;
- instrucciones para modificar registros de administración de memoria y cronómetro;
- instrucciones HALT (parada): el programa usuario no podrá obtener computador;
- instrucciones para activar y desactivar el sistema de interrupciones: ya que el funcionamiento adecuado del cronómetro y las Entradas - Salidas dependen de la capacidad de responder correctamente a interrupciones;
- instrucciones para cambiar de modo usuario a privilegiado;
- instrucciones para modificar el bit de modo del procesador, etc.

Es importante aclarar que el S.O. MS-DOS que se escribió para Intel 8088, no tiene bit de modo, tampoco posee modo dual. Si un programa pierde el control puede destruir al S.O. escribiendo datos encima de él. Varios programas pueden escribir en un dispositivo al mismo tiempo, con resultados desastrosos. Recién en los procesadores posteriores al 80286 Intel incorporó la posibilidad del procesamiento dual.

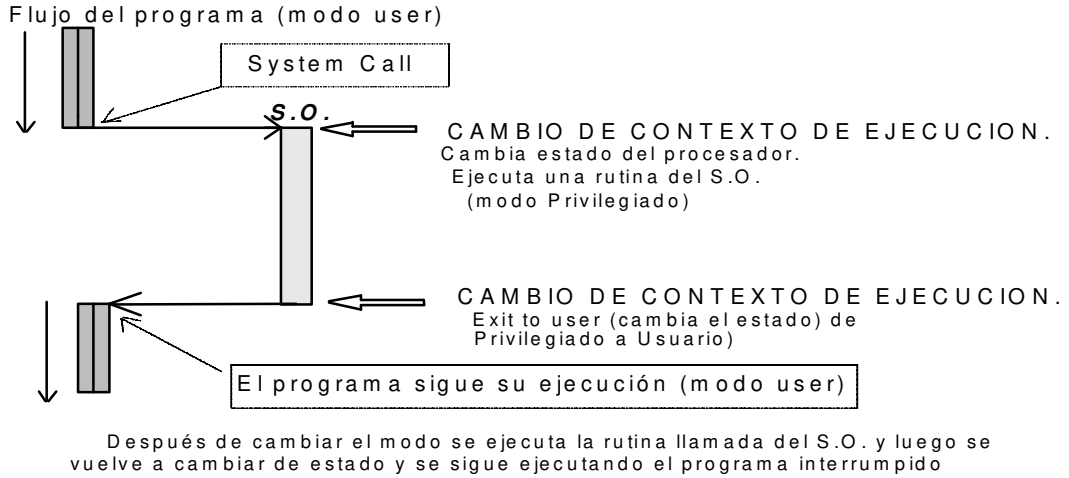


Fig. 1.23 Cambio de contexto en el modo de ejecución

La interacción la realiza el programa en ejecución con el sistema operativo sin intervención del usuario lo hace mediante las **llamadas al sistema**. Cuando se realiza un **System Call** (usaremos los términos System Call, SYSCALL o Llamadas al Sistema, como equivalentes), se cambia el "modo" o estado del procesador de acuerdo a que tipo de instrucciones está ejecutando.

1.3.2. Gestión de Eventos y Entrada Salida:

a) Interrupciones

b) Canales D.M.A. (*Direct Memory Access*).

a) Interrupciones

Se llama interrupción a la detención del programa en ejecución debido a una condición externa al procesador o sea, que el procesador es forzado a reconocer la ocurrencia de un evento en el sistema mediante una señal. En general, el procesador después de preservar los contenidos de todos los Registros y cierta información acerca del estado del proceso (salva en el stack[5] de la memoria central como un VECTOR DE ESTADO o Bloque de Control del Proceso PCB[6]), reemplaza el contenido del contador de programa (Program Counter PC), por una dirección predeterminada donde se encuentra la rutina de interrupción. A continuación, el procesador comienza a ejecutar la rutina de interrupción, que es un programa escrito para responder a la condición generada por el pedido de atención de esa interrupción.

Dependiendo del tipo de interrupciones, intervendrá el S.O. para dar un adecuado tratamiento al evento que generó el Pedido de Atención de la Interrupción.

Recomendamos leer el **Anexo A** de este módulo sobre interrupciones, dado que es muy importante comprender este concepto para entender los próximos módulos.

b) Canales D.M.A. (Direct Memory Access).

El sistema DMA, se ideó, para transmitir datos desde un dispositivo periférico a la memoria RAM o viceversa. En los comienzos, el procesador debía ocuparse de leer los datos del dispositivo y enviarlos a la memoria central, no pudiendo dedicarse a otra actividad mientras estaba ejecutando ese proceso.

Para agilizar esa transferencia, el dispositivo accede directamente a memoria mediante un procesador DMA sin que el procesador central tenga que preocuparse de ese proceso, con lo que se consigue un gran ahorro de tiempo, además de ser más rápido se efectúa concurrentemente.

[5] Snack o pila es un área de memoria central donde se guardan estructuras de datos de control para la ejecución.

[6] Usaremos los términos Process Control Block - PCB-, Vector de Estado o Bloque de Control del Proceso, como equivalentes

El acceso directo a memoria se inventó con el propósito de liberar al procesador de la carga de atender a algunos controladores de dispositivos y realizar la transferencia de datos. Para comprender su funcionamiento vale la pena revisar cómo trabaja un controlador sin DMA. Por ejemplo cuando un proceso requiere leer algunos bloques de un dispositivo, se envía una señal al controlador con la dirección del bloque deseado. El controlador lo recibe a través del 'bus' y el proceso puede estar esperando la respuesta (trabajo sincrónico), o puede estar haciendo otra cosa (trabajo asíncrono). El controlador recibe la señal y lee la dirección del bus. Envía a su vez una o varias señales al dispositivo mecánico (si es que lo hay), y espera los datos. Cuando los recibe los escribe en un buffer local y envía una señal a un árbitro del bus del sistema indicándole que necesita usar el bus para realizar la transferencia. El procesador y el DMA se intercalan en el uso del bus. Cuando el procesador no usa el bus, el DMA comienza a leer y transferir byte por byte o palabra por palabra los datos del buffer del controlador (a través del *device driver*), hasta terminar la operación.

El DMA actúa como un procesador secundario en cuanto a que tiene el poder de tomar el control del 'bus" e indicarle al verdadero procesador que espere. Cuando el controlador tiene listos los datos, el DMA escucha si el bus está libre aprovechando esos ciclos para ir leyendo los datos del buffer del controlador e ir escribiéndolos en el área de memoria que el procesador le indicó. Cuando todos los datos fueron escritos, se le envía una interrupción al procesador para que use los datos. El ahorro con el DMA es que el procesador ya no es interrumpido (aunque sí puede ser retardado por el DMA), salvando así el cambio de contexto y además el DMA aprovechará aquellos ciclos en que el bus no fue usado por el procesador.

El hecho de que los controladores necesiten buffers internos se debe a que conforme ellos reciban datos de los dispositivos que controlan, los deben poder almacenar temporalmente, ya que el procesador no está listo en todo momento para leerlos.

1.4. Tipos de Sistemas Operativos

a) Sistema Monoprocesador o Sistema Multiprocesador. b) Sistema Monousuario. c) Sistema Multiusuarios. d) Sistemas de Propósito General. e). Sistemas de Propósito Especial.

Existen distintas arquitecturas computacionales y por ende, distintos tipos de S.O.. Proponemos una clasificación de ellas:

Según **cantidad de procesadores** en:

- **Sistema Monoprocesador:** un solo procesador
- **Sistema Multiprocesador:** Varios procesadores

Según la **cantidad de Usuarios** que soporta, en:

- **Monousuario**: un solo Usuario.
- **Multiusuarios**: más de un usuario trabajando simultáneamente con el computador.

Una tercera clasificación puede ser hecha en base a las aplicaciones que se ejecutan en un sistema de cómputo. Entonces según sus aplicaciones, los S.O. se pueden dividir en:

1. **De propósito general**: Son los que proporcionan una amplia gama de servicios y deben adaptarse a cualquier ambiente, tipo de aplicaciones, modos de operación, dispositivos, etc.
2. **De propósito especial**: Construidos a medida debido a arquitecturas especiales o aplicaciones con requerimientos especiales como control de procesos industriales.

Veamos cada uno de ellos:

a) Sistema Monoprocesador o Sistema Multiprocesador

Sistemas Monoprocesador. Un sistema operativo monoprocesador es aquél que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil. El ejemplo más típico de este tipo de sistemas es el DOS, algunos Windows y MacOS.

Sistemas Multiprocesador: Son sistemas con más de un procesador, compartiendo memoria y periféricos. Se usan dos aproximaciones. La más común es asignar a cada procesador una tarea

específica. Un procesador central controla el sistema, y los demás tienen tareas específicas, o le piden instrucciones al procesador principal. Este esquema define una relación maestro / esclavo. Ejemplos de estos sistemas son aquellos que usan un Procesador Frontal (Front-End Processor), para manejar lectoras e impresoras a alguna distancia del procesador central. Estos sistemas están compuestos generalmente de un computador grande, que es la computadora principal (Host o Mainframe), y un computador más pequeño que es responsable de la E/S de la terminal.

El otro tipo común de multiprocesamiento es el uso de redes. En éstas, varias computadoras independientes pueden comunicarse, intercambiar archivos e información mediante una Red. Cada computador tiene su propio sistema operativo (pudiendo ser el mismo, o no), y opera independientemente.

Actualmente existen **Sistemas Operativos Distribuidos**, o sea, con características similares a la anterior, pero en donde sus funciones están distribuidas entre distintos procesadores ubicados en áreas geográficas distantes vinculadas mediante redes.

Multiproceso. Un sistema operativo multiproceso se refiere al número de procesadores del sistema, que es más de uno y éste es capaz de usarlos todos para distribuir su carga de trabajo. Generalmente estos sistemas trabajan de dos formas: simétrica o asimétricamente. Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos. Cuando se trabaja de manera simétrica, los procesos o partes de ellos, son enviados indistintamente a cualquiera de los procesadores disponibles, teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema.

Un aspecto importante a considerar en estos sistemas es la forma de crear aplicaciones para aprovechar los varios procesadores. Existen aplicaciones que fueron hechas para correr en sistemas monoproceso que no toman ninguna ventaja a menos que el sistema operativo o el compilador detecte secciones de código paralelizable, los cuales son ejecutados al mismo tiempo en procesadores diferentes.

b) Sistema Monousuario

Estos sistemas se basan en máquinas que admiten a un sólo usuario que utiliza todos los recursos sin compartirlos simultáneamente con otros usuarios. Es el caso de las computadoras personales (P.C.), o estaciones de trabajos específicas (Workstations). Las características básicas de estas máquinas son que los S.O. permiten ejecutar una variada gama de paquetes de software y también el desarrollo y ejecución por parte del usuario de sus propios programas. El énfasis de su diseño está puesto en una interfase amigable entre el usuario y el equipo, un lenguaje de control sencillo al igual que las facilidades de uso de los archivos y periféricos.

Son de interés relativo debido que el estudio de los mismos pertenecen a un caso particular de los sistemas multiusuarios, donde se simplifican sus conflictos y administradores de recursos con soluciones poco complejas.

c) Sistema Multiusuarios.

1) Sistemas de Consultas de Información:

Estos sistemas se caracterizan por las consultas triviales que se efectúan sobre archivos o bases de datos con el objeto de obtener una información dada en un tiempo relativamente corto. Es el caso de Bases de Datos, como ser, médicas, o bibliográficas, etc.

2) Sistemas de Gestión de Operaciones:

Se caracterizan por trabajar sobre una Base de Datos frecuentemente modificada, posiblemente varias veces por segundo, tal el caso de la gestión de operaciones bancarias o reservas de pasajes aéreos. Los principales problemas en éstos sistemas reside en mantener la información permanentemente actualizada y las operaciones simultáneas sobre los mismos datos. Estos problemas los debe resolver el SO.

d) Sistemas de Propósito General:

Los Sistemas Operativos de propósito general se emplean en computadores que soportan una amplia gama de aplicaciones. Estos sistemas están diseñados para mantener un flujo constante de trabajo (Workflow), en forma de tareas a ser ejecutadas por la máquina. Debido al gran número, y diversidad de trabajos, el sistema debe proveer soportes utilitarios y facilidades para manejar una gran cantidad de unidades periféricas. La disponibilidad y el control de éstos utilitarios constituyen, junto con la organización del flujo de trabajo, las principales funciones de un Sistema Operativo de propósito general.

Los Sistemas de propósito general pueden clasificarse en dos grandes grupos:

- Sistemas Batch.
- Sistemas de accesos múltiples (Multiaccess).

Sistemas Batch: se caracterizan por el hecho de que, una vez introducida una tarea en la máquina, el usuario no mantiene contacto con la misma hasta que concluye su ejecución. Hay dos modos de proponer las tareas al sistema. La primera es a través del operador, quien recibe el trabajo y lo pone a ejecutar, cuando finaliza devuelve los resultados de la ejecución al usuario. La otra forma es un Sistema Batch denominado de "entrada remota de trabajos" (en inglés Remote Job Entry (RJE)), que permite ordenar la ejecución de los trabajos a través de dispositivos de Entrada Salida (E/S), que pueden estar en lugares alejados (Remotos), de la máquina.

Sistemas de accesos múltiples: se caracterizan porque el usuario puede iniciar, vigilar, controlar, o suspender la ejecución de su programa desde cualquier terminal del sistema. El S.O. hace que los diferentes trabajos compartan los recursos computacionales de forma que cada usuario tenga la sensación de disponer todo el Sistema en forma exclusiva para él solo.

Muchos S.O. combinan los modos de ejecución de Batch, para trabajos rutinarios que no requieran interactividad (como ser obtención de listados o control de stocks, etc.), y de accesos múltiples, debido a las ventajas que presenta desde el punto de vista interactivo, como por ejemplo, depuración de errores, preparación de documentos, o nuevos desarrollos.

Los Sistemas pueden estar implementados sobre un solo procesador instalado en una sala o sobre varios procesadores, que pueden estar dentro de un equipo (Multiprocesadores), o distribuidos en lugares distantes entre sí, intercomunicados por medio de líneas de transmisión de datos o red. En estos últimos casos, el Sistema Operativo debe, además, coordinar las actividades de los diferentes procesadores o computadores, asegurando que se lleve a cabo un adecuado flujo de información entre ellos.

e). Sistemas de Propósito Especial.

Decíamos que son aquellos especialmente diseñados para cubrir ciertas necesidades específicas. Veamos algunos Sistemas Operativos de propósito especial:

- **S.O. en tiempo real:**

Un sistema operativo de tiempo real se usa generalmente como un dispositivo de control en una aplicación dedicada en que el procesamiento debe realizarse dentro de un tiempo dado. Las principales características que deben brindar son:

1. Garantizar la respuesta a eventos externos dentro de límites de tiempo preestablecidos.
2. Los parámetros más importantes son el tiempo de espera de la entrada, tiempo de procesamiento de la entrada, y un rápido almacenamiento de la misma.
3. Existen dos tipos de Sistemas de Tiempo Real:
    a) Aquellos en que el tiempo de respuesta no es muy crítico: reservas de pasajes, aplicaciones comerciales on-line, control de tránsito vehicular, control de tráfico telefónico, etc.
    b) Los que el tiempo de respuesta es muy crítico (Sistemas estimulados por eventos externos deben generar respuestas a estos eventos), control de procesos industriales, recolección de datos experimentales, etc.

Los esquemas de tiempo real a) y b) se diferencian también por el tratamiento de interrupciones y el manejo de las prioridades.

El sistema en tiempo real incluye el software (que debe organizar, administrar y operar la transferencia de los datos y proveer la adecuada sincronización de tiempos). El hardware está

compuesto por:

- ***sensores***: elementos que detectan mediante una alteración de sus características un cambio en el ambiente en que miden.
- ***transductores***: Traducen un cambio físico en una corriente o tensión eléctrica.
- ***conversores (ADC/DCA):*** Convierten las variaciones de corrientes o tensiones eléctricas en pulsos binarios (Analogic - Digital Converter / Digital - Analogic Converter).
- ***interfases:*** Puertos, Impresores, Graficadores, Registradores, Visualizadores, Señalizadores, Alarmas, Consolas, Teclados, Actuadores, etc., etc..
- ***procesadores:*** Son los responsables de las tareas de control, interrupciones, tiempos, almacenamiento de datos temporales y definitivos, generación de órdenes de control o mandos a distancia, etc.

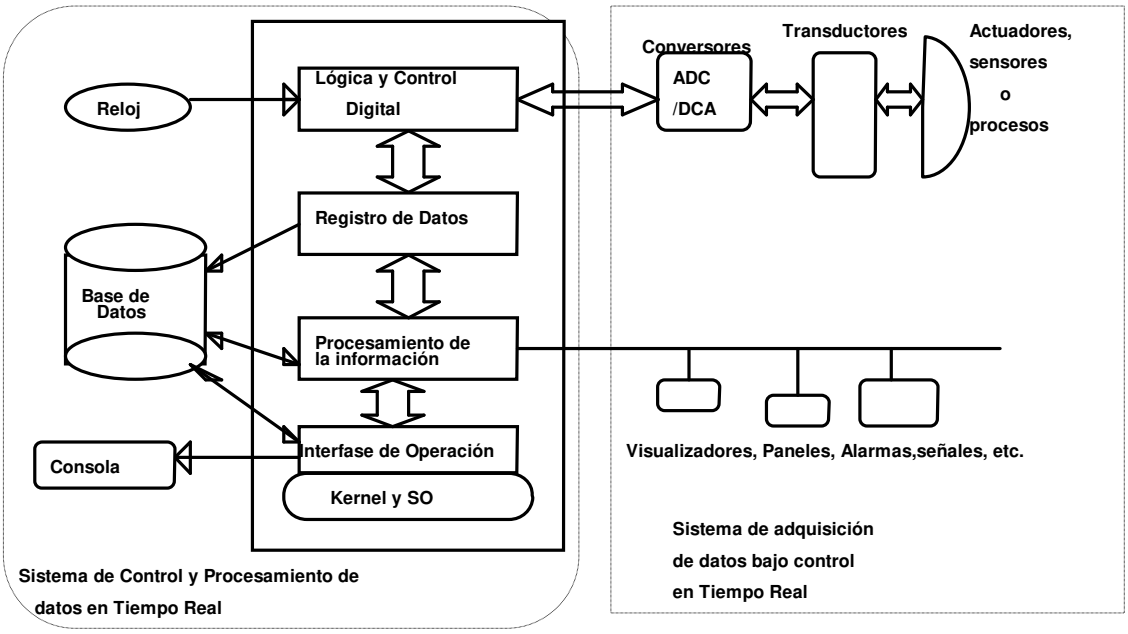


Fig. 1.23 Esquema de un Sistema de Control de Procesos en Tiempo Real

Las características de los procesos, ya sean industriales, médicos, u otros; son que el computador recibe como entrada una información, la procesa y efectúa una acción como consecuencia del proceso. Esta acción se conoce como realimentación (en inglés feed-back), cuyo efecto es mantener la estabilidad del sistema. En algunos de estos procesos se requiere una respuesta de acuerdo a la evolución del fenómeno, entonces se dice que el sistema está trabajando en tiempo real (ejemplo un sistema que monitorea un fenómeno).

Un caso particular de S.O. de Tiempo real son los Sistemas de Control de Procesos.

La principal función del S.O. en un control de proceso es la de proporcionar el máximo de fiabilidad con el mínimo de intervención humana. En caso de una falla del hardware el S.O. debe ser capaz de detener el proceso bajo control o reconfigurar el Hardware para que siga operativo el sistema.

- **S.O. con tolerancias a fallas (*fault-tolerance Operating System*)**

1. Usado en aplicaciones donde se debe proveer un servicio continuo o cuyo mantenimiento es dificultoso o muy costoso.
2. Se suele utilizar un conjunto de redundancias en recursos y chequeos internos.
3. EL S.O. detecta y corrige errores; y recupera el sistema habilitando reemplazos de los componentes en mal funcionamiento o vuelve atrás operaciones que motivaron perdidas de datos.
4. Ejemplos de aplicaciones: Sistemas de seguridad en el área nuclear, sistemas espaciales, cajeros automáticos, bases de datos on-line, etc.

- **S.O. virtuales**

2. Especialmente diseñados para ejecutar varios S.O. (o distintas versiones de uno mismo), concurrentemente en una máquina creando la ilusión de varias máquinas idénticas.
3. Todos los SO trabajan en modo usuario respecto al S.O. virtual, pero están en modo privilegiado con respecto a los programas que corren bajo ese S.O..
4. Ejemplos de aplicaciones: Verificación de un nuevo S.O., migraciones de un Sistema Operativo a una nueva Versión (*release*), conversiones de aplicaciones a un nuevo ambiente.

Conclusiones:

1. Ningún tipo de S.O. es mejor que otro, cada uno presta un servicio especial y constituye un compromiso entre distintas alternativas.
2. Para cubrir los distintos trabajos en un centro de cómputos se debería disponer de distintos Sistemas Operativos, o uno solo que ofrezca una amplia gama de servicios.
3. Los S.O. de propósitos generales son normalmente más complejos y extensos que los especializados.

1.5. Características de diseño de un S.O.: a) Metodología de diseño. b) Principios de diseño, historia y tendencias. c) Criterios de diseño arquitectónico. d) Maquinas Virtuales. e) Multiprocesadores y Microkernel.

En ésta sección se analizarán los distintos ítems a tener en cuenta cuando hay que diseñar un sistema operativo, o cuales son los puntos que se analizaron al diseñar un sistema operativo en general. El estudio de esto nos es útil para poder realizar la comparación de los mismos.

Los diseños difieren en los siguientes puntos:

- Objetivos.
- Restricciones.
- Aplicaciones y ambientes.
- Criterios.
- Razones Económicas.
- Otros

En cuanto al diseño de un S.O. se debe tener fundamentalmente en cuenta los siguientes puntos:

- Conocer el procesador en profundidad (Hardware).
- Conocer ampliamente el Juego (Set), de instrucciones del procesador.
- Conocer profundamente técnicas algorítmicas.
- Definir las características que deben incorporarse en el diseño.

Además los objetivos de diseño de un Sistema Operativo deben cumplir los siguientes requerimientos:

- Simple.
- Portable.
- Estructurado (modular).
- Confiable.
- Soporte de múltiples usuarios o procesos.
- Soporte de red o procesamiento distribuido.
- Etc.

Los principales objetivos que debe incluirse en el diseño, entre otros, son:

- Todo el uso del sistema requiere de eficiencia.
- Tiempo transcurrido entre tareas o procesos.
- Tiempo ocioso del procesador central.
- Tiempo de ejecución utilizado por los procesos, ya sea Batch u otros.
- Tiempo de respuesta en los accesos de los recursos.
- Grado de utilización de los recursos.
- Rendimiento (Trabajos ejecutados por hora).
- Facilidad de uso
- Etc.

Normalmente no todos estos criterios pueden ser satisfechos simultáneamente, esto se debe fundamentalmente a restricciones tecnológicas y de costos. Por ejemplo:

- **Confiabilidad**: Teóricamente un S.O. debería resolver todos los conflictos y contingencias que se generan en el sistema y estar libre de fallas o errores. En la práctica esto nunca

ocurre.

- **Modularidad y documentación**: Un sistema operativo debe estar estructurado por módulos y claramente documentado las interfases a los efectos de facilitar su programación o corrección de los errores, para ello se requiere de un elaborado estilo de programación y de documentación.
- **Tamaño reducido**: el objetivo es que no utilice o monopolice grandes cantidades de recursos restándolos a los requerimientos de la producción debido a su condición de competidor privilegiado frente a los demás.

a) Metodología de diseño:

Lo primero que hay que decidir al diseñar un SO es su finalidad y el tipo de proceso que se quiere realizar a través de él (Batch, Time-Sharing, multiprogramming, etc.).

Para ello es preciso tener en cuenta las necesidades que pueden plantearse:

- **Requisitos de usuario**: Sistema fácil de usar y de aprender, seguro, rápido y adecuado al uso a que se le quiere destinar.
- **Requisitos de software**: Donde se engloban aspectos como el mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a errores, y flexibilidad.

Esto facilita el diseño en cuanto la estructuración de los módulos

Existen autores que son partidarios de un enfoque jerárquico o capas para el diseño de sistemas operativos. En la base de la jerarquía se encuentra el hardware del computador, a veces denominado simplemente *"máquina pura"*. En el siguiente nivel de la jerarquía (o en varios de los siguientes niveles en algunos diseños), están las diferentes funciones del núcleo (Kernel), que se ven como si formaran una *máquina ampliada*, es decir, un computador que ofrece no sólo su lenguaje de máquina como apoyo al sistema operativo y a sus usuarios, sino también un grupo de funciones adicionales proporcionadas por el núcleo. Estas posibilidades adicionales se denominan a menudo *primitivas.*

Arriba del núcleo, en jerarquía, se encuentran los diferentes procesos[7] del sistema operativo que trabajan en apoyo de los procesos de usuario; por ejemplo, los *procesos de administración de los dispositivos*, que se encargan en la práctica de supervisar las operaciones de entrada/salida de los dispositivos del sistema para beneficio de los diversos usuarios. En la cima de la jerarquía se encuentran los procesos de usuario.

Se ha visto que los diseños jerárquicos son más fáciles de depurar, modificar y verificar. En los diseños en que el núcleo está distribuido en varios niveles de jerarquía, elegir qué función colocar en cada nivel requiere un análisis cuidadoso. En tales diseños, con frecuencia sólo se permite hacer llamadas a funciones situadas jerárquicamente por debajo de quién hace la llamada; es decir, cada nivel sólo puede llamar a las funciones que están colocadas en el nivel inmediato inferior.

En los sistemas más recientes existe la tendencia de colocar gran parte del núcleo en micro código. Ésta es una técnica de seguridad efectiva, pues impide la alteración del núcleo y con una cuidadosa codificación se puede lograr que se ejecuten más eficientemente las funciones del núcleo.

b) Principios de diseño, historia y tendencias:

A partir de aquí se intentará integrar a los algoritmos y estructuras de datos, utilizadas para resolver problemas específicos en los Sistemas Operativos, para discutir los problemas de diseño e implementación de sistemas operativos.

OBJETIVOS: El primer problema a enfrentar en el diseño de un sistema operativo, es el de definir los objetivos y especificaciones del sistema.

El nivel más alto de definición estará afectado directamente por:
Selección del hardware (Arquitectura)

- Tipo de sistema (Batch, Tiempo compartido, Monousuario, Multiusuario, Distribuido, Tiempo real, Propósito general)

El segundo problema a enfrentar es el de requerimientos, que pueden ser básicamente divididos

[7] Proceso es un programa (rutina o función) en ejecución con una estructura de datos asociado

en dos grupos:

- Objetivos del usuario
- Objetivos del sistema

Objetivos del Usuario: Los usuarios desean de un sistema propiedades muy obvias:

- Fácil de usar
- Fácil de aprender
- Seguro
- Rápido

Estas especificaciones no son muy útiles en el diseño de un sistema operativo, pues no existe una opinión general de cómo obtener estos objetivos.

Obje**tivos del Sistema:** Un conjunto similar de propiedades pueden ser definidas por las personas que deberán diseñar, crear, mantener y operar un sistema operativo, o sea, el sistema debe ser:

- Fácil de diseñar
- Fácil de implementar
- Fácil de mantener
- Flexible
- Libre de errores
- Eficiente
- Seguro

Nuevamente estos requerimientos son vagos y no tienen una solución general. O sea que no existe una única solución para definir los requerimientos para un sistema operativo. El amplio rango de sistemas existentes hoy en día muestra cómo diferentes requerimientos desembocan en una amplia variedad de soluciones.

Por ejemplo, los requerimientos para un sistema operativo monousuario para microprocesadores son bien diferentes de los de un mainframe como el MVS, un sistema para un alto número de usuarios y multitarea para equipos grandes.

La especificación y diseño de un sistema operativo es una tarea de mucha creatividad. Si bien no existen reglas para resolver este problema, existen algunos principios generales en la Ingeniería de Software que pueden ser aplicables.

MECANISMOS Y POLÍTICAS: Un principio muy importante es la separación de mecanismos y políticas. La mayoría de las discusiones mantenidas hasta ahora se han referido a los mecanismos. Los mecanismos determinan **cómo** hacer algo. Por otra parte las políticas deciden **qué** hacer.

Por ejemplo, un algoritmo de planificación de uso del recurso procesador por prioridades es un mecanismo que determina cómo conmutar ese recurso entre distintos procesos. La definición de prioridades por proceso es la política a implementar.

Este tema se discutirá ampliamente en el capítulo de Protección y Seguridad, donde se comentan que los mecanismos sólo proveen los controles de acceso y las políticas deciden cómo deben ser usados.

La separación de mecanismos y políticas provee mucha flexibilidad. Debería ser tal que un cambio de política pueda ser aplicado sin necesidad de cambiar mecanismos. Por ende, lo deseable es un mecanismo general.

c) Criterios de diseño arquitectónico:

Esto puede considerarse como un diseño en capas o módulos, Máquina Virtual, multiprocesadores, Microkernel o Cliente - Servidor.

EN CAPAS O MÓDULOS: Todo sistema complejo, y un sistema operativo lo es, puede ser creado sólo partiéndolo en piezas pequeñas. Cada una de estas piezas debe estar "bien definida" y se debe prestar especial atención a la definición de entradas, salidas y funciones. Es decir, se habla de modularización, la cual se puede obtener de distintas maneras, pero la más usual en sistemas operativos es el " diseño en capas". Esto consiste en dividirlo en un número de niveles, cada uno de estos niveles debe cumplir una función y mantener comunicación con sus adyacentes, siendo el nivel más alto la interfase con el usuario y el más bajo el hardware. A título de ejemplo, Dijkstra en 1968 utilizó este método para diseñar el THE con las siguientes capas :

- NIVEL 5 Programas de usuario.
- NIVEL 4 Buffering para dispositivos de E/S.
- NIVEL 3 Manejador de consola de operador.
- NIVEL 2 Administración de memoria (Paging, Software).
- NIVEL 1 Planificación de CPU ; (prioridades); Semáforos (P y V).
- NIVEL 0 Hardware.

Tuvo que implementar la paginación por software pues no contaba con los elementos de hardware necesarios. Antes de acceder a una página se verificaba que estuviese en memoria. Implementó el algoritmo LRU[8] para la remoción.

La mayor ventaja de un diseño en capas es la modularidad. Los niveles son seleccionados y pueden pedir funciones sólo a los niveles más bajos. Esto permite que sea más fácil la detección de errores y la verificación de funcionamiento.

Un nivel puede ser analizado e inclusive cambiado sin afectar el resto del sistema. El diseño en capas puede ser utilizado de varias maneras, por ejemplo, en el sistema Venus de Liskov, diseñado en 1972, se utilizó esta modalidad, pero los niveles 0 a 4 estaban puestos en micro código.

- NIVEL 6 Programas de usuario.
- NIVEL 5 Manejadores de dispositivos y planificadores.
- NIVEL 4 Memoria virtual (Segmentación paginada).
- NIVEL 3 E/S, Canales.
- NIVEL 2 Planificación de la CPU (prioridades), y operadores P y V.
- NIVEL 1 Intérprete de instrucciones.
- NIVEL 0 Hardware.

Para gestionar la complejidad de los S.O. y solucionar estos problemas, se ha prestado mucha atención a la estructura del software de los S.O. El software debe ser modular. Esto ayuda a organizar el proceso de desarrollo de software y reduce las tareas de diagnostico y detección de errores.

Para S.O. grandes la programación modular por si sola no es suficiente. En su lugar se usa el concepto de niveles jerárquicos y abstracción de información. La estructura jerárquica de un S.O. moderno separa sus funciones de acuerdo a su complejidad, su escala característica de tiempo y su nivel de abstracción.

Es útil con el fin de obtener una visión general de los S.O.. Presentar un S.O. jerárquico, que consta de los siguientes niveles:

Nivel 1: Circuitos Electrónicos
Consta de circuitos electrónicos, donde los objetos que se tratan son registros, celdas de memoria y puertos lógicos.
Ejemplo de operaciones en este nivel: CLEAR, TRANSFER, ACTIVATE, COMPLEMET. Etc.

Nivel 2: Juego de Instrucciones del Procesador (Instruction Set)
Es el conjunto de instrucciones del procesador que se debe conocer en profundidad.
Ejemplo de operaciones en este nivel: LOAD, STORE, ADD, SABTRACT, BRANCH. Etc.

Nivel 3: Funciones o Procedimientos
Añade el concepto de procedimiento o subrutina, así como las operaciones de llamada y retorno.
Ejemplo de operaciones en este nivel: MARK, STACK, CALL, RETURN. Etc.

Nivel 4: Interrupciones
Introduce las interrupciones, las cuales hacen que el procesador salve el contexto actual e invoque a una rutina de tratamiento de la interrupción.
Ejemplo de operaciones en este nivel: INVOKE, MASK, UNMASK, RETRY. SET_INTERRUPT, RESET_INTERRUPT, EXIT_INTERRUPT, SIGNAL_INTERRUPT, ENABLE_INTERRUPT, DISABLE_INTERRUPT.

Estos primeros cuatro niveles no forman parte del S.O., sino que constituyen al hardware del procesador.

Nivel 5: Primitivas de procesos

[8] LRU son las siglas de Least Recently Used o sea la menos recientemente usada. Lo estudiaremos en el Módulo de Administración de Memoria.

En este nivel se introduce la noción de proceso como programa en ejecución y la capacidad de suspender y reanudar los procesos y los métodos de sincronización, como por Ej. El semáforo.
Ejemplo de operaciones en este nivel: SUSPEND, RESUME, WAIT, SIGNAL, END, ABORT, LOAD, EXECUTE, CREATE_PROCESS, TERMINATE_PROCESS, etc.

Nivel 6: Almacenamiento secundario Local.
Tiene que ver con los dispositivos de almacenamiento secundarios del computador. En este nivel se sitúan las funciones de ubicación de las cabezas de lecturas y escrituras, y se producen las transferencias reales de bloques.
Ejemplo de operaciones en este nivel: READ, WRITE, ALLOCATE, FREE. Etc.

Nivel 7: Memoria Virtual
Crea un espacio de direcciones lógicas para los procesos. Este nivel organiza el espacio de direcciones virtuales. Cuando el bloque necesario no esta en memoria, la lógica de este nivel le solicita una transferencia al nivel 6.
Ejemplo de operaciones en este nivel: READ, WRITE, FETCH. Etc.

Hasta este punto, el S.O. se ocupa de los recursos de un solo procesador.

Nivel 8: Comunicaciones entre procesos
Se dedica a la comunicación de información y mensajes entre procesos. Una de las herramientas más potentes de este nivel es el PIPE, que es un canal lógico para el flujo de datos entre los procesos.
Ejemplo de operaciones en este nivel: CREATE, DESTROY, OPEN, CLOSE, READ, WRITE. Etc.

Nivel 9: Sistema de archivos (File system)
Da soporte al almacenamiento a largo plazo de los archivos con nombre. Los datos del almacenamiento secundario se contemplan en términos de entidades abstractas de longitud variable, en contraste de cómo lo ve el nivel 6, en pistas, sectores y bloques de tamaño fijo.
Ejemplo de operaciones en este nivel: CREATE, DESTROY, OPEN, CLOSE, READ, WRITE. Etc.

Nivel 10: Dispositivos (Devices)
Es el que proporciona el acceso a los dispositivos externo mediante interfaces estandarizadas.
Ejemplo de operaciones en este nivel: OPEN, CLOSE, READ, WRITE. Etc.

Nivel 11: Directorios
Es responsable de mantener la relación entre los identificadores internos y externos de los recursos y objetos del sistema. El ID externo es el nombre que puede usar una aplicación o un usuario. El ID interno es una dirección que es usada por niveles inferiores del S.O. para ubicar y controlar un objeto.
Ejemplo de operaciones en este nivel: CREATE, DESTROY, ATTACH, DETACH, SEARCH, LIST. Etc.

Nivel 12: Procesos del Usuario
Da el soporte para todo lo concerniente al manejo ordenado de los procesos. Esto incluye el espacio de direcciones virtuales del proceso, una lista de objetos y procesos con los que puede interactuar y las limitaciones de dicha interacción, los parámetros pasados en el momento de la creación.
Ejemplo de operaciones en este nivel: QUIT, FORK, KILL, SUSPEND RESUME. Etc.

Nivel 13: Shell.
Ofrece al usuario la interfase con el S.O., es decir es el shell.
Ejemplo de operaciones en este nivel: SENTENCIAS EN LEGUAJE DE SHELL (SCRIPTS)

d) Maquinas Virtuales:

Una aplicación interesante del diseño en capas es el del concepto de Máquinas Virtuales. El sistema operativo VM (Virtual Machine), de IBM es el mejor ejemplo.

Usando técnicas de planificación de procesador y de memoria virtual, un sistema operativo

llamado CP (Control Program de IBM), puede crear la ilusión de múltiples procesos ejecutando en su propio procesador dentro de su propia memoria (virtual). Los procesos necesitan *"llamadas al sistema"* y un *"file system"* que no están provistos por el hardware *"llano"*. Las máquinas virtuales, o sea el CP, no proveen funciones adicionales, pero proveen una interfase que es idéntica a la del hardware llano. Cada proceso es provisto de una copia virtual de la computadora.

Los recursos físicos son compartidos para crear las máquinas virtuales. El planificador de procesos es utilizado para compartir procesador y hacer creer a cada proceso / usuario que posee su propio procesador. La paginación por demanda puede proveer a cada procesador virtual su propia memoria virtual. De hecho la memoria virtual para una máquina virtual puede ser mayor o menor que la memoria física de la computadora real. El spooling y la administración de información (el CP más otro sistema operativo ejecutado en la máquina virtual), proveen lectores e impresoras virtuales.

Un usuario común de tiempo compartido se transforma así en un operador de consola de una máquina virtual.

Sin embargo existen dificultades en el sistema de discos. Si existen más máquinas virtuales que discos disponibles, aparentemente esto no sería posible. La solución es entregar a cada máquina virtual un disco virtual, que es idéntico al disco verdadero salvo a lo que hace en su tamaño. Esta fracción de disco entregado a cada máquina virtual se denomina "minidisco". Luego, un usuario puede ejecutar en su máquina virtual cualquier software que desee, u otros sistema operativos, siempre y cuando respeten la arquitectura virtual provista por las máquinas virtuales.

En los sistemas IBM se carga normalmente un sistema operativo llamado CMS (*Conversational Monitor System*). Esta forma de trabajo puede proveer una fácil partición al problema de diseño de un sistema interactivo multiusuario en dos trozos pequeños. Aún cuando esta es una buena solución trae aparejado una serie de problemas que consisten básicamente en proveer un duplicado exacto de la máquina física. Recordemos, que una máquina física trabaja en dos modos, modo usuario/esclavo o modo Kerne/monitor.

En este diseño, VM, el software que ejecuta dentro de una máquina virtual puede ejecutar en modo monitor, ya que es un sistema operativo, pero la máquina virtual ejecuta en modo usuario, ya que es un proceso (para el CP). En consecuencia se tendrá un modo usuario y un modo monitor, ejecutando ambos dentro de una máquina virtual que ejecuta en modo usuario. Las acciones que causen una transferencia de modo usuario a modo monitor en una máquina real deben causar también la transferencia de modo usuario virtual a mono monitor virtual dentro de una máquina virtual.

Veamos cómo opera: cuando se realiza una llamada al sistema, hecha por un programa ejecutando en modo usuario virtual, ésta causa una transferencia al monitor de máquina virtual (CP), en la máquina real. Cuando el CP toma el control, cambia los contenidos de registros y PC (o PSW – ***P**rogram **S**tatus **W**ord-),* de la máquina virtual para simular el efecto de la llamada al sistema. Luego deja a la máquina virtual en modo monitor virtual. Si la máquina virtual desea realizar una operación privilegiada, como una operación de E/S, y como está trabajando en modo usuario real, esta instrucción es atrapada por el monitor de máquinas virtuales (CP), y le simula su efecto. La mayor diferencia que se aprecia es en el tiempo, ya que si las instrucciones deben ser interpretadas tardarán mucho más.

En el caso del VM de IBM esto trabaja bastante bien, pues todas las instrucciones, salvo las privilegiadas son ejecutadas directamente sobre el hardware y solamente las privilegiadas son simuladas. El concepto de máquina virtual tiene algunas ventajas, cada máquina está completamente aislada de las demás, luego no existe problema de la seguridad, pero tampoco hay compartición de recurssos. Para proveer compartición hay que compartir minidiscos, lo que es controlado por software. También es posible definir una red de máquinas virtuales, que también son controladas por software.

Otra de las grandes ventajas de las máquinas virtuales es la posibilidad de probar todo tipo de software, inclusive sistemas operativos sin afectar al resto de la producción, ya que cualquier problema queda circunscrito a la máquina virtual.

e) Multiprocesadores y Microkernel:

Los sistemas operativos para sistemas multiprocesadores buscan obtener de las máquinas seguridad y mayor capacidad de procesamiento. La mayor capacidad de procesamiento se obtiene,

obviamente, al disponer de mayor cantidad de procesadores sobre los cuales distribuir procesos, tareas o instrucciones de acuerdo a tablas de procesos, estado de ocupación de recursos y tablas de administración de la información. Además deberán proveer el soporte a las primitivas de generación de procesos, como así también el control de los mismos.

En el caso de seguridad lo que se busca es que dos procesadores, uno primario y uno secundario ejecuten en forma simultánea lo mismo, de tal manera que si el primario falla, el secundario al no recibir señales del primero se haga cargo de la continuación de las tareas (Ej. Sistema Tandem con el S.O. Guardian).

MICROKERNEL: Es un diseño interesante pues se colocan pocas funciones en el Kernel:

- Comunicaciones o Mensajes entre procesos
- Multiprogramación
- Manejo de Interrupciones
- Y algunos servicios de E/S y Administración de Memoria.

El resto se resuelve como funciones de servicio externas al kernel.

IMPLEMENTACIÓN: Tradicionalmente los sistemas operativos fueron escritos en lenguaje ensamblador. Actualmente se han adoptado lenguajes de alto nivel que permiten expresar en forma cómoda las funciones del sistema operativo, reservando solo la escritura en assembler de las partes más sensibles y que requieran cuidado en su desempeño. Tenemos casos como el Master Control Program (MCP), de Burroughs escrito en Algol; Multics, desarrollado por el MIT escrito en PL/1 y Unix o Linux escrito en lenguaje C. Solamente unas 900 líneas del Unix están escritas en assembler como el despachador (dispatcher), y algunos manejadores de dispositivos (Device Handlers).

Las ventajas de usar un lenguaje de alto nivel son la de escribir el código más rápido, más compacto y más fácil de entender y corregir. Sus desventajas pueden estar dadas en la velocidad de ejecución y espacio requeridos. Pero mejorando los compiladores este problema se resuelve. De todas maneras las partes críticas como el manejador de páginas, el planificador y el despachador pueden ser reemplazadas si se detecta que se transforman en "cuello de botella".

GENERACIÓN DE SISTEMAS: Los sistemas operativos son especificados para utilizar una gran variedad de máquinas y de dispositivos. Cuando se instala en un determinada computadora es necesario configurarlo o generarlo para esta computadora. Este proceso se lo conoce como Generación de Sistemas (SYSGEN). En este paso hay que indicar:

- Tipo de procesador a ser usado
- Cantidad de memoria real disponible
- Dispositivos disponibles
- Opciones del sistema operativo deseado

Esta información puede ser incluida en el sistema operativo, de tal manera que sea necesario compilarlo todo nuevamente, o linkeditar una tabla con estas opciones o utilizarlos como tabla de parametrización en el momento de iniciar las actividades del sistema operativo. Esto se conoce como Configuración del Sistema.

En los S.O. modernos estos reconocen el Hardware disponible y automáticamente se configuran instalando todo lo necesario para que el sistema funcione óptimamente.

1.6. Características de los S.O.. a) Características comunes a todos los S.O.. b) Gestión y reparto del conjunto de recursos. c) Gestión de la Información. d) Cooperación entre los procesos. e) Protección.

En general, se puede decir que un Sistema Operativo tiene las siguientes características funcionales:

- Conveniencia: Un Sistema Operativo hace más conveniente el uso de una computadora.
- Eficiencia: Un Sistema Operativo permite que los recursos de la computadora se usen de la manera más eficiente posible.
- Habilidad para evolucionar: Un Sistema Operativo deberá construirse de manera que permita el

desarrollo, prueba o introducción efectiva de nuevas funciones del sistema sin interferir con el servicio del mismo.

- Encargado de administrar el hardware: El Sistema Operativo se encarga de manejar de una mejor manera los recursos de la computadora en cuanto a hardware se refiere, esto es, asignar a cada proceso una parte del procesador para poder compartir los recursos.
- Relacionar dispositivos (gestionar a través del kernel): El Sistema Operativo se debe encargar de comunicar a los dispositivos periféricos, cuando el usuario así lo requiera.
- Organizar datos: para acceso rápido y seguro.
- Manejar las comunicaciones en red: El Sistema Operativo permite al usuario manejar con alta facilidad todo lo referente a la instalación y uso de las redes de computadoras.
- Procesamiento por bytes: de flujo a través del bus de datos.
- Facilitar las entradas y salidas: Un Sistema Operativo debe hacerle fácil al usuario el acceso y manejo de los dispositivos de Entrada / Salida de la computadora.
- Técnicas de recuperación de errores: Debe proveer las rutinas necesarias para protegerse, y poder recuperarse de los errores (ya sean de hardware, o de software), con el menor perjuicio para los usuarios.
- Evita que otros usuarios interfieran: Debe evitar que los usuarios se bloqueen entre ellos, informándoles si esa aplicación esta siendo ocupada por otro usuario.
- Generación de estadísticas: Para poder *cobrarle* a los diferentes usuarios del centro de costos la parte proporcional que les corresponde del gasto generado por el mantenimiento del mismo.
- Permite que se pueda compartir: tanto el hardware como los datos de los usuarios y entre los usuarios.

a) Características comunes a todos los S.O.

En general los cuatro problemas comunes que deben tratar y resolver cualquier S.O. son:

- Gestión y reparto del conjunto de recursos.
- Designación de objetos y acceso a la información de esos objetos.
- Cooperación entre procesos paralelos también llamados concurrentes.
- Protección para preservar la integridad de los recursos y los usuarios.

Veamos cada uno de ellos en forma explícita.

b) Gestión y reparto del conjunto de recursos:

Cuando hablamos de Recursos nos referimos a: Tiempo de procesador, E/S, reloj, Memoria Central (MC), Datos, Variables, Archivos, etc.

Distinguimos dos tipos de recursos: físicos y lógicos. Ejemplo procesador, Memoria, Tiempos, Variables.

En cuanto a su uso, distinguimos dos tipos de recursos:

- **No Compartible**: Uso restringido a un solo Proceso. Ej.: Impresora, Unidad de Cinta, procesador (aunque a primera vista el procesador parecería ser un recurso compartido, en realidad no lo es, ya que en un instante dado, solo un proceso puede estar usándolo. Lo que se hace para simular que es un recurso compartido es multiplexarlo en el tiempo), etc.
- **Compartibles:** empleados por varios procesos en forma concurrente. Ej.: Áreas de Memoria Central con rutinas re-entrantes, Archivos de Lectura-escritura, etc.

La Concurrencia puede verse como la activación de varios procesos a la vez. Supongamos:
n (procesadores), y **m** (procesos). Si **n = m** y **m < n** (no hay Problemas), pero si **m > n** (si hay problemas), entonces hay que multiplexar (conmutar), los recursos entre los procesos, por ejemplo, durante un dado intervalo de tiempo.
Si se lo ve desde una perspectiva de escala de tiempo mayor, alternan distintos procesos en un mismo procesador. Esto se conoce como **multiplexado** del procesador entre los procesos.

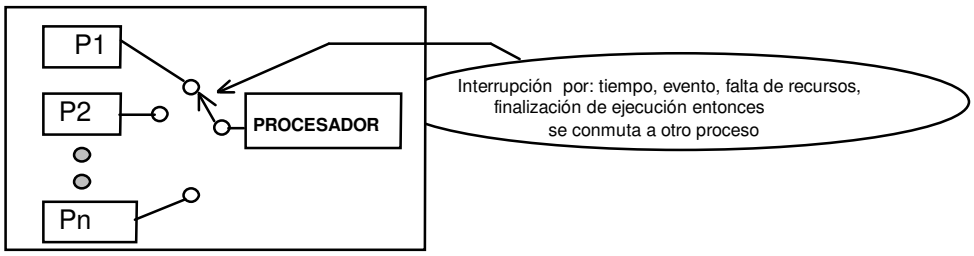


Fig. 1.24. Multiplexación del procesador

Para asignar los recursos disponibles a los trabajos de acuerdo a las políticas de diseño del S.O., los objetivos principales son:

- Mejorar el uso del Hardware.
- Satisfacer a los usuarios (Optimizando el tiempo de respuesta y servicios, etc.)

Existen dos estrategias para la asignación de los Recursos:

- Estática. Ejemplo COBOL (Assign)
- Dinámica. Ejemplo PASCAL (Write)

La figura 1.25 nos presenta la problemática de los recursos compartidos por varios usuarios, ya que se genera una gran demanda por la competencia ejercida por esos escasos recursos.

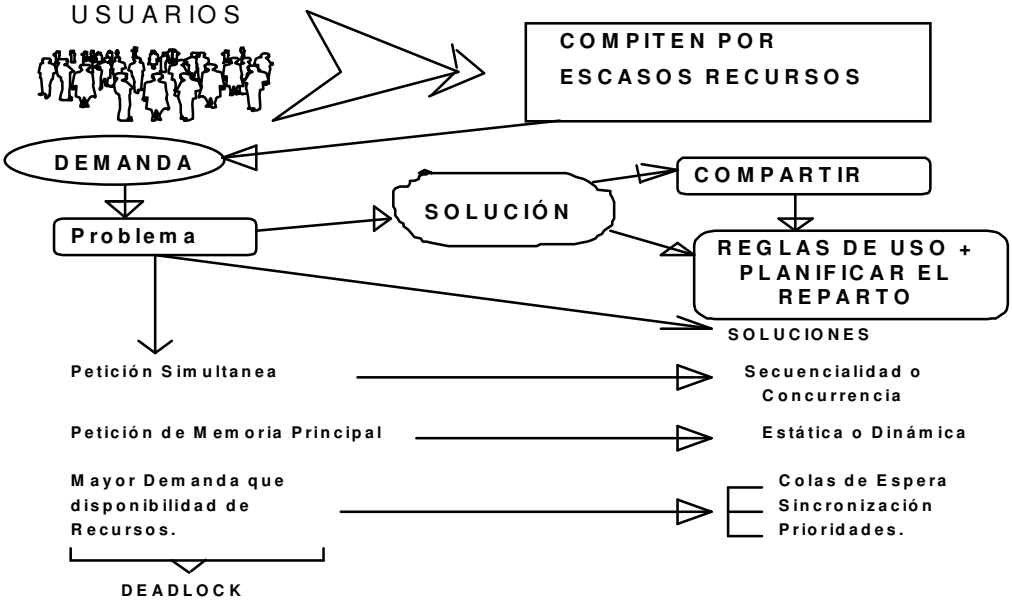


Fig. 1.25. Concepto de reparto de recursos y sus conflictos

Peligro: pérdida o degradación de las prestaciones o bloqueos entre procesos. La solución al problema generado es del tipo político, pues deben compartir el uso de esos recursos demandados. Para poder hacerlo sin conflictos se requiere planificar el reparto, fijar claras reglas de uso y asignarle la función de autoridad al S.O., quién entregará el recurso a quién corresponda y lo recuperará nuevamente para redistribuido luego. Esta función de árbitro neutral en los conflictos es necesaria para una correcta gestión de caracter administrativa que debe ejercer el Sistema Operativo.

c) Gestión de la Información:

Los objetivos de la Gestión de la Información son los siguientes:

- **ENLACE (Binding):** Asocia el objeto a los ***espacios físicos*** que puede direccionar un Procesador (para consultarlo, modificarlo, operarlo, etc.), sin ambigüedades.
- **ASIGNACIÓN DE DESCRIPTORES (DELAY BINDING TIME):** Permite retrasar la elección de

un recurso.

- **MANEJO DE DIRECCIONES (VIRTUALES o REALES):** Ejemplo: Traslación estática o Dinámica (en Segmentación - Paginación que lo veremos en el Módulo 5).

En la siguiente figura se ilustra los objetivos planteados y en particular la vinculación (Binding), de identificadores de objetos durante el procesamiento de los programas.

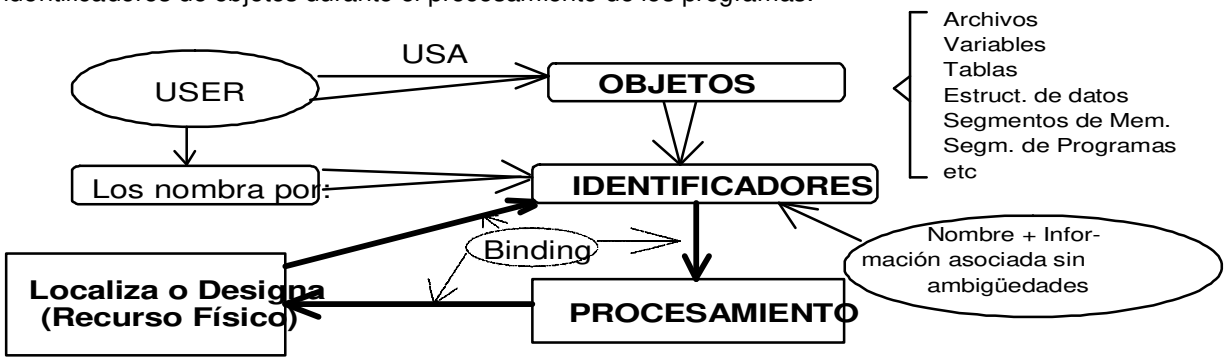


Fig. 1.26. Gestión de la información de los objetos

d) Cooperación entre los procesos:

Los procesos que interactúan generan conflictos en los siguientes casos, cuando:

- Compiten por el uso de los Recursos.
- Cooperan para alcanzar el objetivo de ejecutar las tareas del Usuario.
- Comparten objetos.
- Se Comunican entre ellos.

Estos problemas, generalmente, se plantean en los accesos a los recursos escasos en donde se visualizan los conflictos por lo que se solucionan a través de herramientas de sincronización y de comunicaciones ofrecidos por el S.O. Estas Herramientas se estudiarán en el módulo 4.

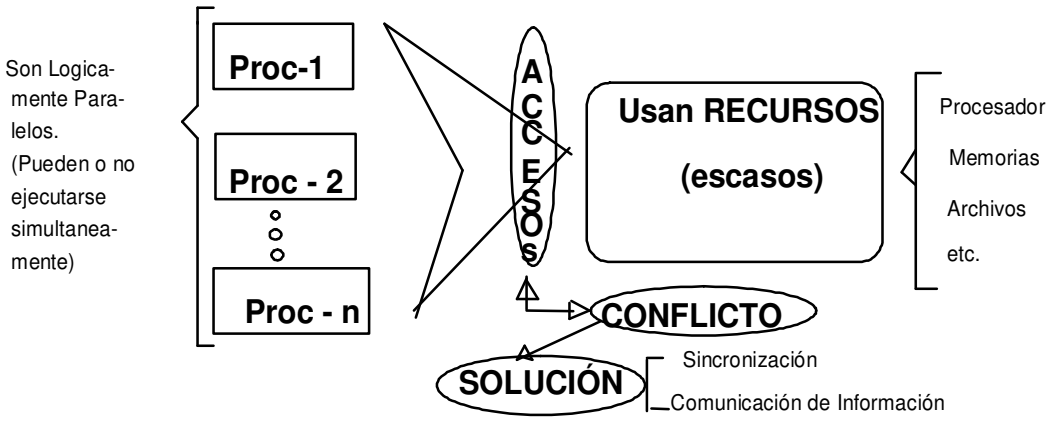


Fig. 1.27 Concepto de cooperación entre los procesos y sus conflictos

e) Protección:

El objetivo de la protección es el de garantizar la integridad de los recursos (léase recursos físicos como así también información), y los procesos, como así también a los usuarios entre sí.

Decíamos que para resolver este conflicto el S.O. ofrece los siguientes mecanismos:

1. **Ejecución dual de instrucciones**: Maestro - Esclavo
2. **Mutua Exclusión**: consiste en asegurar que los recursos compartidos sean accedidos por ***un solo***

proceso a la vez bajo ciertas condiciones (lo veremos más detallado en el módulo 4 de Sincronización y Comunicación entre procesos.

3. **Control de accesos**: Permitir a los usuarios acceder solamente a los recursos mínimos y necesarios para que puedan realizar correctamente su trabajo, pero no permitirle que accedan a todo el centro de cómputos si no lo necesitan.

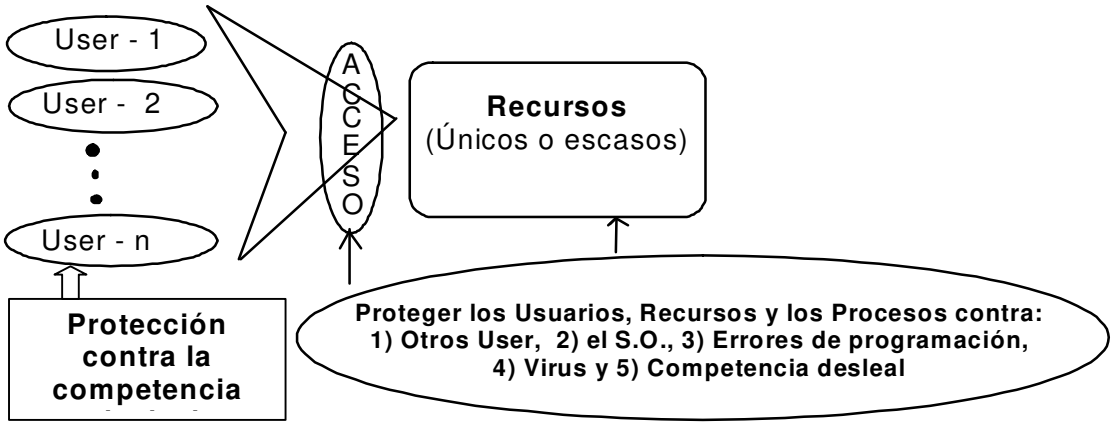


Fig. 1.28 Concepto de Protección

1.7. Arquitectura (Estructura), de un Sistemas Operativos

a) Estructura Tradicional o Monolítica. b) Estructura en Estratos o Jerárquica. c) Estructura cliente / servidor. d) Máquinas virtuales.

El sistema operativo se divide lógicamente en pequeños módulos y se crea una interfase bien definida para cada módulo. Para esto se debe definir e interrelacionar cuidadosamente su función, sus entradas y sus salidas.

Generalmente cuando se mencionan las arquitecturas o estructuras de los sistemas operativos se refieren a la forma en que se organizan los distintos módulos componentes del mismo, en estratos o niveles. Básicamente los S.O. tienen dos interfases: una con el Hardware (llamada Kernel o Núcleo), y otra con los usuarios y sus programas (llamada Shell).

Presentaremos las siguientes cuatro estructuras (también llamadas Arquitecturas de S.O.).

a) Estructura Tradicional o Monolítica

Es la estructura de los primeros SO constituidos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra como se observa en la figura siguiente

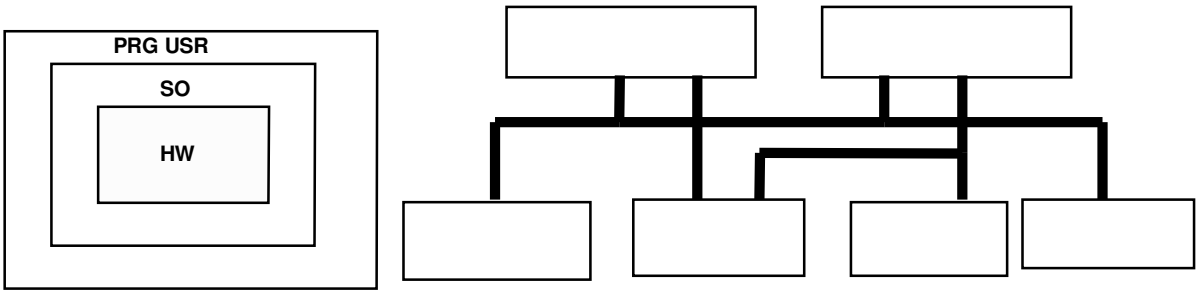


Fig. 1.29. Estructuras tradicionales o monolíticas de un S.O.

Los siguientes puntos caracterizan a esta estructura:

- Interfases y niveles de funcionalidad no están bien separadas.

- No tienen estructura interna definida. Ejemplo: Aplicaciones pueden acceder a rutinas básicas de E/S para escribir directamente en la pantalla y unidades de disco.
- El SO es un conjunto de rutinas que se llaman entre sí libremente.
- No existe modularización, ni ocultamiento de la información.
- Difícil la reconfiguración y actualización.

Características generales:

- Carecen de protecciones y privilegios.
- Existe una mínima estructura. Servicios del sistema se piden a través de llamadas especiales (**supervisor call** o **System call**).
- Ejemplos: MS-DOS, las primeras versiones de UNIX y la mayoría de los Sistemas Operativos para micro-computadoras.

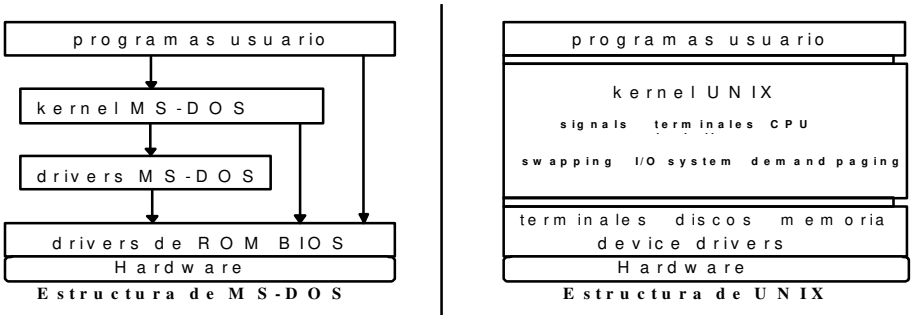


Fig. 1.30 Estructuras tradicionales o monolíticas de los S.O. MS-DOS y UNIX

A continuación en la figura se ilustran éstas estructuras algo más detalladamente. El Kernel proporciona la planificación del procesador y otras funciones del S.O. mediante las llamadas al sistema que utilizan los programas del usuario cuando se ejecutan.

- **MS-DOS** se escribió para el procesador Intel 8088. No ofrece modo dual de operación ni protección del hardware. Los diseñadores tuvieron que dejar accesible el hardware base. Fue diseñado e implantado por unas cuantas personas que no tenían la menor idea de la popularidad que tendría. Se escribió para lograr la mayor funcionalidad en el menor espacio.
- **El UNIX** original, inicialmente estaba limitado por la funcionalidad del hardware. Éste se encuentra divido en dos partes una de ellas comprende los programas del sistema y la otra el kernel. El kernel se encuentra divido en drivers de dispositivos y en las interfases

Fig. 1.31 Interfases del Shell y del Kernel

Los programas del sistema, que hacen de interfase con el usuario, utilizan las llamadas al sistema proporcionadas por el Kernel para ofrecer funciones útiles como por ejemplo la compilación, y de esta forma el hardware ejecuta los requerimientos del Usuario.

Fig. 1.32 Definición de interfases del S.O.

b) Estructura en Estratos o Jerárquica

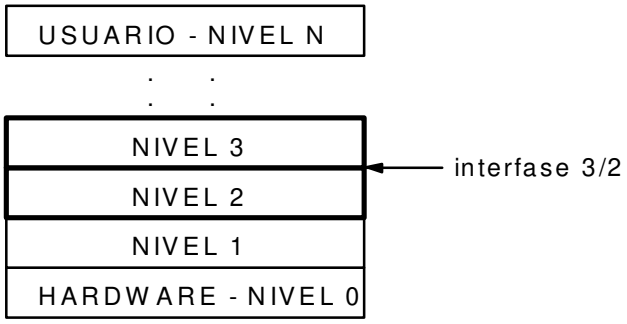


Fig. 1.33 Estructura en estratos o jerárquica de un S.O.

Esta estructura se basa en:

- El principio de ordenamiento jerárquico.
- Los distintos módulos se organizan en una jerarquía de niveles.
- El módulo de nivel N funciona usando los servicios provistos por el nivel N-1.
- Cada módulo no sabe ni necesita saber como se implementan los servicios del módulo inferior, sólo conoce la interfase. Se oculta la existencia de ciertas estructuras de datos, operaciones y hardware a niveles superiores.
- El S.O. mantiene un mayor control sobre computadoras y aplicaciones que lo utilizan.
- Se usa un enfoque descendente, así, de esta manera se determinan funciones y características globales y se las separa en componentes.

El sistema es dividido en módulos, utilizando un "enfoque por capas", que separa al S.O. en varias capas (niveles), cada una construida sobre las inferiores. Donde cada capa constituye una implantación de un objeto abstracto que contiene datos y operaciones. Estas operaciones solo pueden manipular los datos en ese nivel.

Ventajas:

a) *Facilita la protección*. Cada capa consiste en alguna estructura de datos y un conjunto de rutinas que pueden ser invocadas por las capas de niveles superiores y a la vez esa capa invocar operaciones de capas inferiores (las rutinas no se invocan libremente y sin orden como en la estructura monolítica).

b) Permite implementar el principio de ocultamiento de la información *information hiding*, que facilita a los programadores realizar rutinas de bajo nivel, siempre que las interfases externas permanezcan sin cambios y la rutina lleve a cabo la tarea indicada.

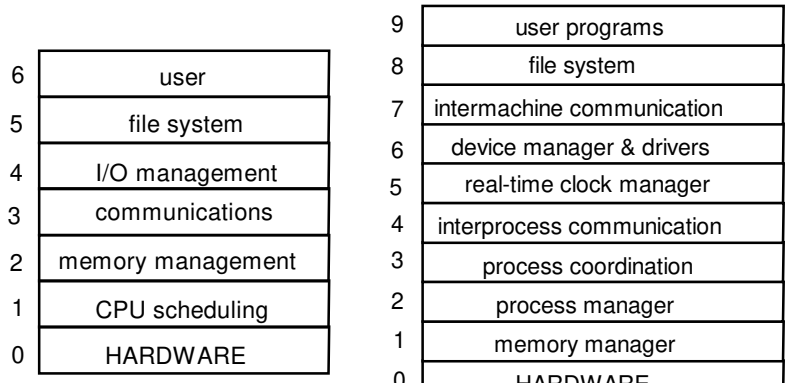
c) Facilita la *sustitución y verificación de componentes* (*modularización*). Las capas se seleccionan de manera que cada una utilice funciones (operaciones), y servicios de las capas inferiores. Así se logra la depuración de todo el sistema. La explicación es lógica: el primer nivel se depura sin preocuparse por el resto del sistema ya que solo utiliza el hardware. Se pasa al segundo nivel pues se sabe que el primero fue depurado y está correcto. Si aparece algún error, debe ser del nivel que se está depurando pues los inferiores ya no presentan errores.

Desventajas:

a) La *definición de los distintos niveles* es difícil. Se debe realizar una planificación cuidadosa. Ejemplo: el manejador de dispositivo para memoria auxiliar debe estar disponible su información para las rutinas de administración de memoria, ya que éstas requieren la utilización de la memoria auxiliar, etc.

b) Algunas comunicaciones son entorpecidas por la jerarquía. Por ej.: si la administración de la CPU necesita un acceso a disco se encuentra en un nivel inferior al administrador de Entrada - Salida, entonces debe pasar por los servicios de otros administradores que se encuentran en el medio.
c) Son más lentos.

Ejemplos:



Sistema Operativo THE
Technische Hogeschool Eindhoven
E.W. Dijkstra (Holanda, 1968)

Sistema Operativo XINU
Douglas Comer (1988)

Fig. 1.34 Ejemplos de S.O. con estructura en estratos.

Otro ejemplo lo constituye el S.O. OS/2, un descendiente directo del MS-DOS, el cual fue creado para superar las limitaciones de éste y agregar multitarea y operación de modo dual, etc. Al adicionar complejidad y tener un hardware mucho más potente se implantó al S.O. en forma de capas.

Otra forma de representar muy similar a ésta, es la denominada **anillos concéntricos** (**rings**), basados en:

- Cada **ring** tiene una apertura (*trap*), por donde se accede desde las capas superiores a las inferiores.
- Las capas internas son más privilegiadas en protección que las externas.

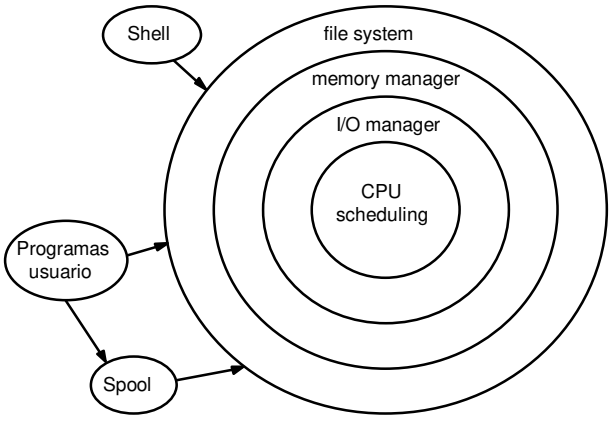


Fig. 1.35. Otra forma de representación en capas de la estructura por estratos.

La solución actual de los S.O. de estructuras jerárquicas es diseñar menos capas pero con mayor funcionalidad, ofreciendo las mayorías de las ventajas del código modularizado a la vez que se evitan los difíciles problemas de la definición e interacción de las capas.

El THE (Technical Hoeschool Eindhoven de Dijkstra E.W.), fue uno de los primeros sistemas operativos que emplearon esta técnica. Dispone de una estructura jerárquica totalmente ordenada en la que cada nivel sólo puede acceder al nivel inferior más inmediato. Este sistema ha sido adoptado por algunos diseñadores de sistemas de control para robots. Aunque en este tipo de sistemas se necesita

acceder directamente desde niveles superiores a los inferiores, pues, por ejemplo debe moverse el brazo del robot desde el nivel de ejecución automática y desde el nivel de programación manual que suele estar en los estratos superiores de la jerarquía. En consecuencia, conviene emplear otra jerarquía, menos restrictiva, en la que cada nivel pueda relacionarse con cualquiera de los inferiores.

Los niveles de jerarquía pueden definirse utilizando un método descendente Top Down (cuando se conoce el problema), o ascendente Bottom Up (cuando no se dispone de una definición exacta del problema o no se dispone de la máquina sobre la que se va ejecutar el S.O.).

c) Estructura Cliente / Servidor

Se basa en lo mismo que el resto de los S.O. convencionales en cuanto el Kernel y los Procesos. Pero presenta grandes diferencias en la forma de distribuir los trabajos entre sus distintas partes.

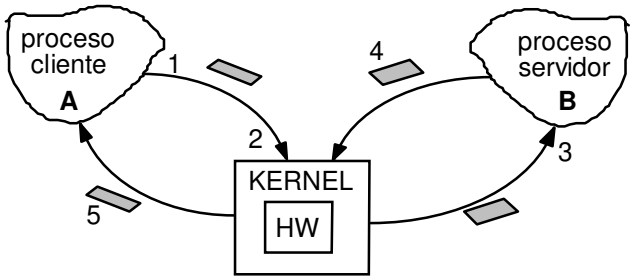


Fig. 1.36 Concepto de un S.O. basado en Cliente - Servidor

Características generales:

- Se remueve la mayor cantidad posible del código del S.O. dejando un Kernel mínimo (microkernel architectures)
- Las Funciones del SO son implementadas como programas usuario.
- Los servicios se efectúan mediante la técnica de *message passing.*

La secuencia para el esquema de la Figura anterior es:

1. El Proceso cliente (A), envía mensaje solicitando un servicio (por ejemplo: lectura de bloque en disco).
2. El Kernel recibe el mensaje, toma las decisiones de planificación correspondientes y lo envía al proceso servidor (B).
3. El Proceso servidor recibe el mensaje enviado por el Kernel y ejecuta la función solicitada (en éste caso lee el bloque del disco).
4. El Proceso servidor devuelve un mensaje al Kernel con el resultado de la operación.
5. Nuevamente el Kernel recibe el mensaje de respuesta y lo reenvía al proceso cliente indicando que el servicio se ha cumplido con éxito o no.

Las funciones del Kernel son pocas. Entre ellas las más importantes son:

- Tratamiento de interrupciones.
- Multiprogramación.
- Sincronización de mensajes.

Ventajas:

a) El *sistema es altamente modular*: porque los componentes son más chicos y manejables.
b) Los distintos *módulos del sistema no tienen acceso directo al hardware:* Esto significa que un error en un servidor no tira abajo al Sistema de cómputo sino solo al servidor en cuestión.
c) Son especialmente útiles en ambientes distribuidos.

Desventajas:

a) Algunos módulos del S.O. no pueden ser implementados como procesos usuario. Ejemplo: drivers de dispositivos, ciertas partes del administrador de memoria (*memory Manager*).

Ejemplos:

- QNX (diseñado en Canadá).
- Mach.

d) Máquinas Virtuales.

Sus principios se basan en:

- El mismo principio que las estructuras en estratos ya presentado.
- En lugar de proveer una visión simplificada del hardware, el S.O. crea *máquinas virtuales*: varias copias **idénticas** del hardware base.
- Cada una de estas máquinas virtuales tiene todas las características de la CPU real (memoria, interrupciones, I/O ports, etc.).
- A cada proceso se le otorga una copia (virtual), del computador subyacente.
- El S.O. de IBM llamado V.M. es el mejor ejemplo de este concepto de máquinas virtuales.
- Por lo tanto, en cada máquina virtual se puede ejecutar cualquier S.O. que funcione en el hardware real.
- Separan dos conceptos que suelen estar unidos en el resto de los sistemas:
  1. La multiprogramación.
  2. La máquina extendida.
- Su objetivo es el de integrar distintos S.O. dando la sensación de ser varias máquinas diferentes.
- El núcleo de estos SO se denomina **monitor virtual**. Realiza la multiprogramación presentando tantas máquinas virtuales como se soliciten.

Estas máquinas virtuales no son máquinas extendidas sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un S.O. diferente, que será el que ofrezca la máquina extendida al usuario.

Ventajas:

a) Cada usuario del sistema puede usar un S.O. distinto.
b) Permite un alto nivel de protección.
c) Todas las máquinas virtuales son independientes.
   Si bien algunos recursos del computador se comparten para crear las máquinas virtuales, dichos recursos no se comparten directamente. Para lograr esto, existen dos estrategias:
   - **Minidisco:** Sigue el modelo de un disco físico compartido, pero se pone en práctica mediante software (con esta técnica los archivos se pueden compartir).
   - **Red de máquinas virtuales:** Cada una puede enviar información a través de una red virtual de comunicaciones. Esta red tiene como modelo a la red física de comunicaciones, pero se implanta en software.
d) Son especialmente útiles para diseño y desarrollo de nuevos Sistemas Operativos.

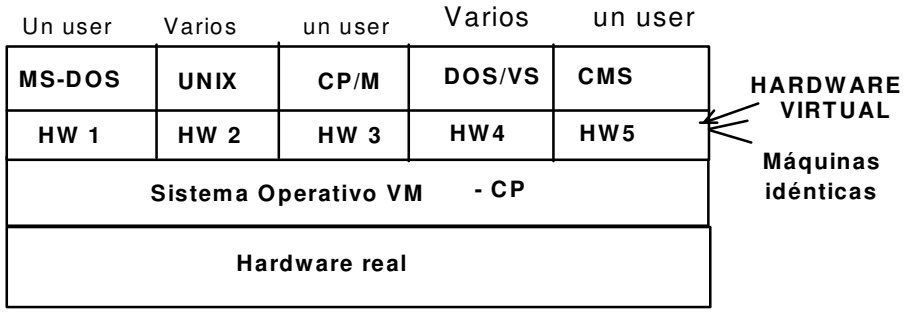


Fig. 1.37. Concepto de Maquina Virtual

A los programadores se les entrega una máquina virtual y el desarrollo se lleva a cabo en ella, no en la máquina física. Pocas veces tienen que interrumpir el funcionamiento del sistema para su desarrollo. De esta forma se evita que un cambio en un punto ocasione errores ocultos en alguna parte, creando una situación que puede ser muy peligrosa en el funcionamiento normal del centro de cómputos

por el poder que tiene el sistema operativo. Como el S.O. se ejecuta en modo monitor o Kernel, por ejemplo, una modificación equivocada de un puntero *(pointer*), puede provocar un error que destruya todo el sistema de archivos o algún otro módulo, por eso es necesario que se pruebe con mucho cuidado las modificaciones que se hacen al S.O., sea este nuevo o actualmente en uso.

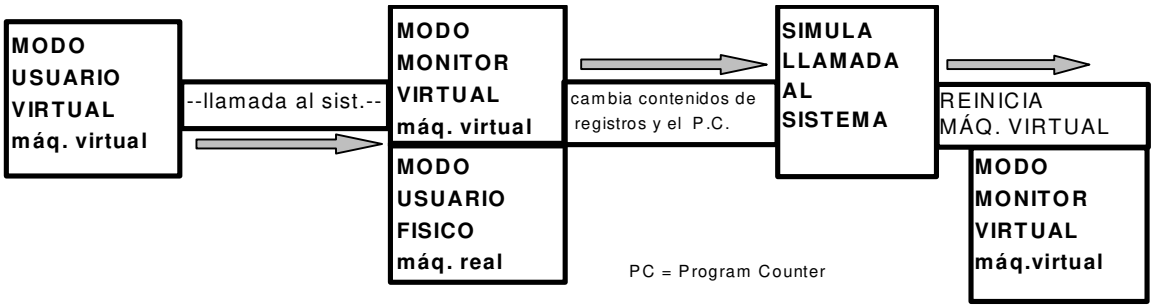
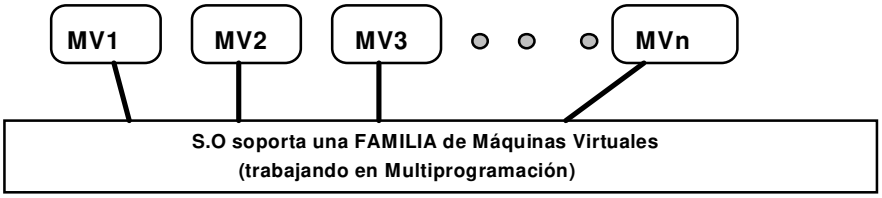


Fig. 1.38. Distintos modos en el concepto de Máquina virtual

Desventajas:

a) Difícil Implementación. Se necesita un duplicado exacto del hardware subyacente.
Por ejemplo, si la máquina física tiene dos modos, también debe poseerlo la máquina virtual. Teniendo en cuenta lo siguiente:
- El software de la máquina virtual puede ejecutarse en modo monitor o Kernel;
- La máquina virtual sólo puede ejecutarse en modo usuario;
- La máquina virtual debe poseer los dos modos;
- Una transferencia de modo Kernel a usuario en máquina real, debe provocar dicha transferencia en una máquina virtual.

b) La simulación del hardware real es muy costosa en recursos.
Por ejemplo si una máquina física tiene tres manejadores de disco pero quiere dar servicio a siete máquinas virtuales; no puede asignarse una unidad de disco a cada una, pero el software de la máquina virtual necesita un espacio considerable en disco para ofrecer memoria virtual y spoolers. En este caso se ha podido dar solución mediante discos virtuales, idénticos en todos los aspectos menos en el tamaño.
En el V.M. de IBM se los llama minidisco. El sistema implanta cada minidisco asignándole la cantidad de pistas que requiera en los discos físicos. La suma de los tamaños de los minidiscos debe ser menor que la cantidad de espacio disponible en el disco físico.



Multiprogramación del S.O.:
- **Administración de Recursos del Sistema**
- **Interpretación de Instrucciones virtuales**

Un S.O. define varios Lenguajes mediante los cuales se definen REGLAS de administración de recursos y solicitudes de servicios.

Fig. 1.39 El S.O. soportando una familia de máquinas virtuales

c) Cada máquina virtual es más lenta que la máquina real.
La CPU se multiprograma entre varias máquinas virtuales, lo que las hace más lentas.
Frente al problema de diseñar un sistema interactivo multiusuario en dos partes más pequeñas en el S.O. V.M., el usuario ejecuta CMS, un sistema interactivo monousuario. El software de la máquina virtual se ocupa de la multiprogramación de varias máquinas virtuales en la máquina física, pero no necesita considerar algún software de ayuda al usuario.

1.8. Componentes mínimos de un S. O.:

a) Primer nivel: Interfases de usuario y de programación (El shell). b) El segundo nivel; (Gestión de procesos, Gestión de la memoria, Gestión de la entrada/salida, Sistemas de archivos. c) El tercer Nivel: Interfase con el hardware (Kernel o núcleos). d) El kernel de UNIX. e) Kernel de un S.O. de TIEMPO REAL.

El S.O. generalmente se divide en tres niveles de administradores: Uno que contempla una serie de administradores de los pedidos (Administración y gestión de los pedidos, Administración de Programas, Administración de Procesos, Administración de Usuarios), que normalmente se engloba en un solo módulo llamado Job Scheduler o Shell, El segundo nivel contempla cuatro o cinco módulos de acuerdo a los distintos tipos de recursos que debe administrar, como ser: Entrada - Salida, Memoria Central, Procesadores, etc. y el tercer nivel le corresponde al Kernel o Núcleo, el Sistema de Acceso de Comunicaciones y el Sistema de Multiprocesamiento.

Un S.O. proporciona el entorno dentro del cual se ejecutan los programas. Para construir este entorno, dividimos lógicamente al mismo en pequeños módulos y creamos una interfase bien definida para éstos programas.

Internamente los S.O. varían en su estructura, organizándose de acuerdo con diferentes esquemas o Arquitecturas. Si bien cada S.O. varía en estructura con respecto a otros, en general se dividen en los siguientes componentes o administradores:

- Interfases de usuario y de programación (El shell).
- Administración de los procesos en ejecución, su coordinación, sincronización e intercomunicación.
- Administración de la Memoria Central.
- Administración de dispositivos de E/S.
- Administración de datos e información del almacenamiento secundario (File Management).
- Administración de recursos (Pedidos, conflictos, devoluciones, etc.).
- Programas de servicio y utilitarios.
- Administración de redes (Networking), protección del Sistema, Intérprete de comandos, etc.
- Administración de programas.
- Interfase con el hardware (Kernel o núcleos).

Todos estos componentes son cuidadosamente organizados en una estructura, ya sea en capas o niveles, para definir claramente las interconexiones entre ellos.

Decíamos que el Sistema Operativo es un conjunto integrado de programas y rutinas que permiten controlar las operaciones de la computadora con el mínimo de intervención humana. Optimiza el empleo de los recursos disponibles, automatiza el flujo de trabajos, toma decisiones sobre el manejo del sistema en la escala de tiempos de la computadora, o sea, a gran velocidad (de un millón a cien millones de veces más rápido que el ser humano).

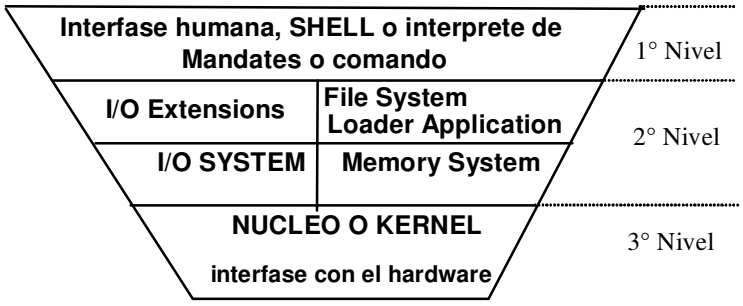


Fig. 1.40 Esquema de un Sistema Operativo

El S.O. tiene programas o rutinas residentes en memoria (Kernel o núcleo, que se carga al

encender la máquina, con el Bootstrapping o I.P.L.), el resto se encuentra en el disco del sistema. Al núcleo del sistema también se lo llama residente, monitor, supervisor, Kernel, etc., debido a que es uno de los programas principales que componen el S.O..

Normalmente está compuesto por módulos o rutinas que controlan los recursos físicos y lógicos de un computador, como ser: Procesadores, Memoria Central y Secundaria, Datos, Archivos, Dispositivos Periféricos, etc.

Los compiladores, cargadores, debuggers y otros módulos del Software de base (utilitarios), usualmente no se consideran parte de los S.O. al igual que los sistemas de Administración de Bases de Datos y el software de control de comunicaciones. En cambio el sistema de acceso de comunicaciones y los programas de servicios son fuertemente dependientes del S.O. y de sus interfases.

a) El primer nivel: Las Interfases de usuario y de programación (El shell)

Es la porción del S.O. que se encarga de hacer de interfase Hombre - Máquina y de transmitir al S.O. los requerimientos del usuario al sistema. La interfase humana es la que nos permite manipular el sistema y es la parte visible del S.O. y puede ser intercambiable. El Shell se puede programar mediante **Scripts.**

Las interfases con la máquina dependen del hardware. Las instrucciones del S.O., se ejecutan a través de Primitivas (*Systems Calls*). Entendemos por "Primitivas" aquellas instrucciones pertenecientes a un lenguaje de programación que se ejecutan completas, sin ser interrumpidas o divididas (*atomic action*).

Decíamos que en el Primer Nivel se tiene un módulo que se ocupa de administrar los pedidos, de la carga de los programas, de crear procesos, de los accesos de usuarios, de las protecciones, etc., y que generalmente a este módulo se lo llama job scheduler o Planificador de trabajos, o Shell, y es quien decide a qué trabajo asignarle el estado de " listo para la ejecución". En algunos sistemas grandes, este módulo se complementa con un lenguaje (Job Control Language), que permite programar las actividades del computador.

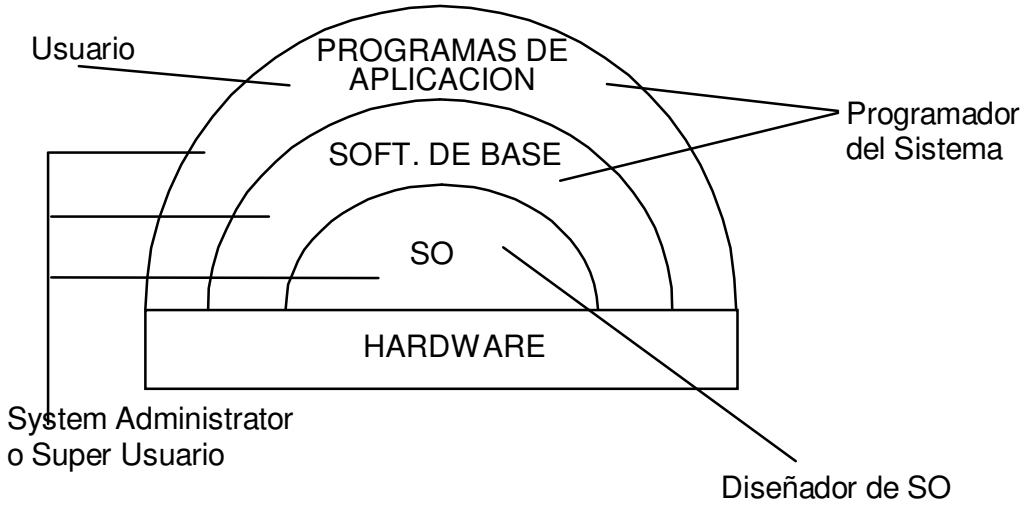


Fig. 1.41 Distintos usuarios y sus visiones de un S.O.

La comunicación con el usuario es a través de algún lenguaje de comandos u órdenes. Esta Interfase con el usuario presenta distintas visiones del S.O. según que usuario este definido en el sistema.

En la mayoría de los casos se definen dos tipos de usuarios y el S.O. provee dos vistas. Una para el Usuario común (User), y otra para el administrador del sistema de cómputo (System Manager o Super User).

En sistemas, donde se desarrollan otras tareas de diseño de software de base o de S.O., existen dos casos más (el Programador del Software de Base y el Diseñador del S.O.), que son poco frecuentes y pueden verse como usuarios especiales que desarrollan su programación como una aplicación particular.

Una visión más generalizada de un sistema de procesamiento de datos y los distintos profesionales que trabajan con él lo vemos en la figura 1.41.

a) Visión del Usuario

El S.O. provee la interfase que proporciona una visión global y abstracta del computador ocultando su complejo funcionamiento interno.

Provee una función muy importante, que es, proteger el uso de los recursos entre él o los usuarios y el sistema para garantizar la integridad y el adecuado reparto de los mismos.

El Shell es un conjunto de comandos u órdenes que ofrecen una gran variedad de servicios para que el usuario pueda ejecutar sus tareas en el sistema de cómputo (por ejemplo copiar o renombrar archivos, ejecutar diversos programas, etc.).

La interacción del usuario con el sistema de cómputo siempre se realiza a través del Shell, el cual transmite al S.O. los requerimientos del usuario al sistema (en algunos sistemas, al Shell se lo llama Administrador de Trabajos o Job Scheduler).

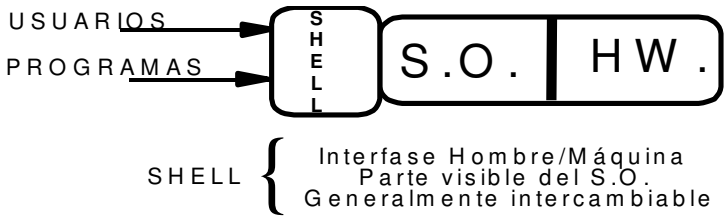


Fig. 1.42 Interfase externa (Shell), de un S.O.

Existen dos grandes **tipos de Shell.** Uno provee una interfase basada en comandos u órdenes llamada **C.L.I.** (**C**ommand **L**ine **I**nterface), que permite ingresar un conjunto de caracteres que el S.O. interpretará como comandos u órdenes y la segunda conocida como **G.U.I.** (**G**raphical **U**ser **I**nterface), que permite accesos a través de iconos o gráficos. Ejemplos: Korn, C, Bourne, Progman, Command, etc.

Entonces, desde el punto de vista del usuario, el principal objetivo del ocultamiento del hardware se debe a:

- **ABSTRACCIÓN**: Proveer una interfase dando una visión global y abstracta del computador de fácil uso y ocultando su funcionamiento.
- **SEGURIDAD**: Protegiendo el uso de los recursos entre los usuarios y el sistema.

b)Punto de vista del System Manager (Administrador de Recursos del Sistema):

Un Sistema Operativo es por excelencia el administrador de todos los recursos ofrecidos por el hardware para alcanzar un eficaz rendimiento de los mismos en cuanto a su uso.

Los Recursos fundamentales son: Procesadores, Memorias, Entrada - Salida, Información, Comunicaciones, etc.

El System Manager configura al S.O. para que el Sistema funcione eficientemente y los usuarios puedan compartir los recursos. Es el responsable de que el Sistema de cómputo funcione con el adecuado rendimiento y con óptimo grado de seguridad.

El System Manager también suele llamarse Superusuario (Superuser o root), en UNIX.

b) El segundo nivel; (Gestión de procesos, Gestión de la memoria, Gestión de la entrada/salida, Sistemas de archivos .

En el Segundo nivel se tiene básicamente los siguientes administradores:

- **Administración de memoria** (Memory Manager): Se encarga de asignar a los procesos la memoria necesaria para su ejecución. Lo estudiaremos en el módulo 5.
- **Administración de periféricos** (I/O System): Comprenden todos los módulos necesarios para la utilización de los periféricos. Ejemplo: asignación de dispositivos, spooler, controladores, etc. Lo estudiaremos en el módulo 6.
- **Administración de información** (File System): Se refiere a las rutinas que permite manipular y manejar el sistema de archivos, permite accederlos (leerlos, grabarlos, crearlos, destruirlos, copiarlos, renombrarlos, etc.). Lo estudiaremos en el módulo 7.
- **Administrador de comunicaciones** (Communication Manager): Es el responsable de compartir los recursos distribuidos mediante una red de computadoras.

c) El tercer Nivel: Interfase con el hardware (Kernel o núcleo)

Es un conjunto de Códigos intensamentes usados por todos los programas en el más bajo nivel como si fuera una extensión de la máquina. Por ello, debe ser desarrollado en lenguaje de máquina y muy optimizado para que su ejecución no produzca un gran sobrecarga (overhead).

Todas las operaciones en las que participan procesos son controladas por la parte del sistema operativo denominada núcleo (nucleus, core o kernel). El núcleo normalmente representa sólo una pequeña parte de lo que por lo general se piensa que es todo el sistema operativo, pero es el código que más se utiliza. Por esta razón, el núcleo reside por lo regular en la memoria central, mientras que otras partes del sistema operativo son cargadas en la memoria central sólo cuando se necesitan. Los núcleos se diseñan para realizar “el mínimo” posible de procesamiento en cada interrupción y dejar que el resto lo realice el proceso apropiado del sistema, que puede operar mientras el núcleo se habilita para atender otras interrupciones.

Contiene básicamente:

1. Una extensión del Juego (Set), de Instrucciones del Procesador.
2. Una extensión del mecanismo de secuenciamiento del Hardware (Dispatcher o Switcher).
3. Un manejador de Interrupciones.
4. Un mecanismo de acceso a memoria central.
5. Una extensión de llamadas para el tratamiento de errores.
6. Un mecanismo de Protección.
7. Un mecanismo de Comunicación entre Procesos.
8. Un mecanismo de tiempo (Timer).
9. Una extensión de E/S.
10. Otros System Calls.

OBJETIVO: Constituir un entorno adecuado en el que se puede desarrollar los distintos procesos.

Es un componente del S.O. que se encarga de:

1. Las Interfases con el Hardware (instruction set del microprocesador).
2. Administrar las interrupciones y excepciones.
3. La Multiprogramación (Low Scheduler).
4. Las Comunicaciones entre procesos.
5. La Gestión del Hardware (Mecanismos de Protección de Accesos a Memoria Central, Mecanismo de Interrupciones, Manejo de Instrucciones Kernel y Usuario, Reloj de Tiempo Real).

Según la arquitectura, el Kernel tendrá más o menos las siguientes características:

1. Los S.O. difieren significativamente en el tamaño y la naturaleza de sus Kernels (Dependen de las aplicaciones y de los criterios de diseño).
2. Las Funciones del Nivel cero de la estructura de los S.O. tienen los siguientes atributos:
   a) Son Residentes en la Memoria Central (Razones de accesibilidad en el menor

tiempo posible).

b) Deben ejecutarse en modo Supervisor (también llamado modo Kernel), como primitivas no interrumpibles (Atómicas).
c) Deben ejecutarse en el más alto nivel de prioridad debido a la Protección.
d) Depende de la máquina y está escrito en el lenguaje ensamblador del Procesador.
e) Generalmente no más de 500 instrucciones (muy optimizado).

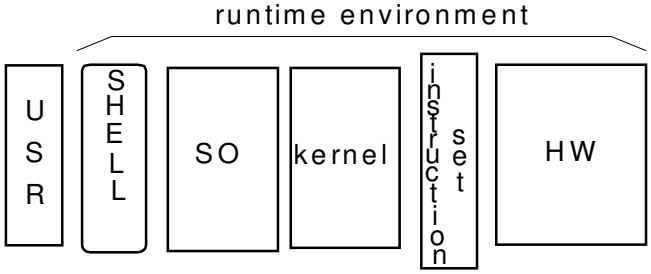


Fig. 1.43. Interfases del S.O.

El núcleo de un sistema operativo normalmente contiene el código necesario para realizar las siguientes funciones:

1. Manejo de interrupciones.
2. Creación y destrucción de procesos.
3. Cambio de estado de los procesos.
4. Despacho.
5. Suspensión y reanudación de procesos.
6. Sincronización de procesos.
7. Comunicación entre procesos.
8. Manipulación de los bloques de control de procesos.
9. Apoyo para las actividades de entrada/salida.
10. Apoyo para asignación y liberación de memoria.
11. Apoyo para el sistema de archivos.
12. Apoyo para el mecanismo de llamada y retorno de un procedimiento.
13. Apoyo para ciertas funciones de contabilidad del sistema.

El Kernel consiste en la parte principal del código del sistema operativo, el cual se encargan de controlar y administrar los servicios y peticiones de recursos y de hardware con respecto a uno o varios procesos, este se divide en 5 subniveles:

- ***Subnivel 1: Gestión de Memoria:*** que proporciona las facilidades de bajo nivel para la gestión de memoria Central y secundaria necesaria para la ejecución de procesos.
- ***subnivel 2:*** Administración del procesador: Se encarga de activar los quantums de tiempo provistos por el reloj (timer), para cada uno de los procesos, creando interrupciones de hardware cuando no son respetadas.

Consta básicamente de dos módulos:

i. Planificador de procesadores y procesos (*Dispatcher*): Decide, una vez llegado un trabajo dividido en procesos, a qué procesador asignar el proceso que tiene que ser ejecutado y le otorga el uso del procesador (no ordena la cola).
ii. Controlador de tráfico (*Traffic Controller*): Este se encarga de crear, modificar y actualizar el contexto asociado a un proceso para posibilitar su pasaje entre los diferentes estados. Por ejemplo, al iniciarse un proceso, previamente pasa por el controlador de tráfico, el cual genera las respectivas actualizaciones, la asignación de memoria y recursos para su ejecución.

- ***Subnivel 3: Entrada/Salida:*** Proporciona las facilidades para poder utilizar los dispositivos de E/S requeridos por los procesos.
- ***Subnivel 4: Información o Aplicación o Interprete de Lenguajes:*** Facilita la comunicación con los lenguajes y el sistema operativo para aceptar las ordenes en cada una de las

aplicaciones. Cuando se solicita la ejecución de un programa, el software de este nivel crea el ambiente de trabajo e invoca a los procesos correspondientes.

- ***subnivel 5: Control de Archivos:*** Proporciona la facilidad para el almacenamiento a largo plazo y manipulación de archivos con nombre, va asignando espacio y acceso de datos en memoria.

Los núcleos de los sistemas operativos se pueden ubicar en dos categorías: monolíticos o micronúcleos (microkernels). El primer tipo de núcleo es el tradicionalmente usado, mientras que los micronúcleos forman parte de las tendencias modernas en el diseño de sistemas operativos.

Para comprender mejor qué diferencias existen entre ambas categorías, se necesita revisar algunos conceptos.

Núcleo Monolítico: Los núcleos monolíticos generalmente están divididos en dos partes estructuradas: el **núcleo dependiente** del hardware y el **núcleo independiente** del hardware. El núcleo dependiente se encarga de manejar principalmente las interrupciones del hardware, hacer el manejo de bajo nivel de memoria y discos y trabajar con los manejadores (Handlers) de dispositivos de bajo nivel. El núcleo independiente del hardware se encarga de ofrecer las llamadas al sistema, manejar los sistemas de archivos y la planificación de procesos. Para el usuario esta división generalmente pasa desapercibida. Para un mismo sistema operativo corriendo en diferentes plataformas, el núcleo independiente es exactamente el mismo, mientras que el dependiente debe reescribirse o configurarse.

Microkernel: La *arquitectura* micronúcleo es aquélla que contiene únicamente el manejo de procesos y threads (hilos), el manejo bajo de memoria, y también da soporte a las comunicaciones y maneja las interrupciones y operaciones de bajo nivel de entrada-salida. En los sistemas operativos que cuentan con este tipo de núcleo se usan procesos *servidores* que se encargan de ofrecer el resto de servicios (por ejemplo el de sistema de archivos), y que utilizan al núcleo a través del soporte de comunicaciones.

Este diseño permite que los servidores no estén atados a un diseño en especial, incluso el usuario puede escoger o programar sus propios servidores. La mayoría de los sistemas operativos que usan este esquema manejan los recursos de la computadora como si fueran objetos: los servidores ofrecen una serie de *llamadas* o *métodos* utilizables con un comportamiento coherente y estructurado. Otra de las características importantes de los micronúcleos es el manejo de threads. Cuando un proceso está formado por un solo thread, éste es un proceso normal como en cualquier sistema operativo.

Los usos más comunes de los micronúcleos es en los sistemas operativos que intentan ser distribuidos, y en aquellos que sirven como base para instalar sobre ellos otros sistemas operativos. Por ejemplo, el sistema operativo AMOEBA intenta ser distribuido y el sistema operativo MACH sirve como base para instalar sobre él otros S.O. Por ejemplo: DOS, UNIX, etc.

d) El kernel de UNIX

UNIX es el kernel (núcleo), de una serie de sistemas operativos de tiempo compartido. Es un programa que siempre está residente en memoria y entre otros, brinda los siguientes servicios:

- Controla los recursos del hardware
- Controla los dispositivos periféricos (discos, terminales, impresoras, etc.).
- Permite a distintos usuarios compartir recursos y ejecutar sus programas.
- Proporciona un sistema de archivos que administra el almacenamiento de información (programas, datos, documentos, etc.).

En un sentido más amplio, UNIX abarca también un conjunto de programas estándar, como pueden ser:

- Compilador de lenguaje C *(gcc).*
- Editor de texto *(vi).*
- Intérprete de órdenes *(sh, ksh, csh).*
- Programas de gestión de archivos y directorios (cp, rm, mv, mkdir, rmdir, etc.).

Los niveles dentro de la arquitectura de UNIX:

El nivel más interno no pertenece realmente al sistema operativo, si no que es el hardware, la

máquina sobre la que está implementado el sistema y cuyos recursos gestiona.

Directamente en contacto con el hardware se encuentra el *kernel* del sistema. Este *kernel* está escrito el lenguaje Ensamblador o C en su mayor parte, aunque coexistiendo con código ensamblador.

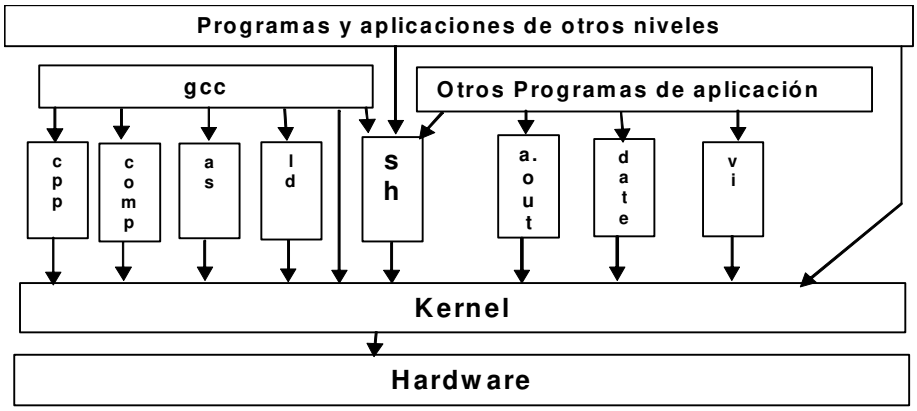


Fig. 1.44. Los niveles dentro de una arquitectura UNIX.

En el tercer nivel de la estructura UNIX se encuentran los programas estándar de cualquier sistema UNIX *(vi, grep, sh, who, etc.)*, y programas generados por el usuario (*a.out* programa ejecutable estándar creado por defecto por el compilador g*cc,* o cualquier otro programa ejecutable). Hay que hacer notar que estos programas del tercer nivel nunca van a actuar sobre el hardware de forma directa. Por lo tanto, debe existir algún mecanismo que nos permita indicarle al *kernel* que necesitamos operar sobre un determinado recurso hardware. Este mecanismo es lo que se conoce como llamadas al sistema *(system calls*) que va a proveer el servicio requerido.

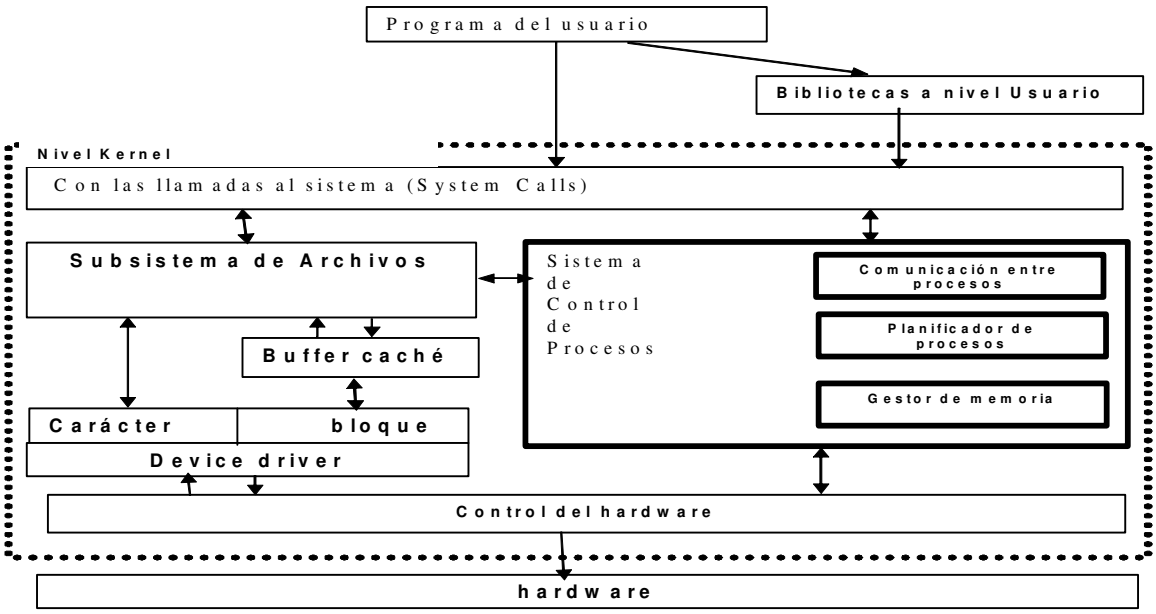


Fig. 1.45. Diagrama de bloque de un kernel UNIX.

Por encima del tercer nivel están las aplicaciones que se sirven de otros programas ya creados para llevar a cabo su función. Estas aplicaciones no se comunican directamente con el kernel.

Los bloques funcionales básicos de que consta el kernel de UNIX: Las llamadas al sistema y su biblioteca asociada representan la frontera entre los programas del usuario y el kernel. La biblioteca asociada a las system calls es el mecanismo mediante el cual podemos invocar un system call desde un

programa C y se encuentra en el directorio /usr/lib/libc.a. Las llamadas al sistema se ejecutan en modo kernel (para muchos microprocesadores o modo supervisor), y para entrar en este modo hay que ejecutar una sentencia en código máquina.

El kernel está dividido en tres subsistemas principales como se observa en la figura 1.45:

- **Subsistema de Archivos:** controla los recursos del sistema de archivos y tiene funciones como reservar espacio para los archivos, administrar el espacio libre, controlar el acceso a los archivos, permitir el intercambio de datos entre los archivos y el usuario, etc.
- **Subsistema de control de procesos**: es el responsable de la planificación de los procesos (scheduler), su sincronización, comunicación entre los mismos (IPC - inter process comunication), y del control de la memoria central. Al módulo gestor de memoria (swapping), se le conoce también como swapper. El planificador o scheduler se encarga de gestionar el tiempo de CPU que tiene asignado cada proceso. El scheduler entra en ejecución en cada quantum de tiempo y decide si el proceso actual tiene derecho a seguir ejecutándose (esto depende de la prioridad y de sus privilegios). La comunicación entre procesos puede realizarse de forma asíncrona (señales), o síncrona (colas de mensajes, semáforos).
- **El módulo de control del hardware**: es la parte del kernel encargada del manejo de las interrupciones y de la comunicación con la máquina. Los dispositivos pueden interrumpir al procesador mientras está ejecutando un proceso. Si esto ocurre, el kernel debe reanudar la ejecución del proceso después de atender a la interrupción. Las interrupciones no son atendidas por procesos, sino por funciones especiales, codificadas en el kernel, que son invocadas durante la ejecución de cualquier proceso.

  Resumiendo el Kernel se ocupa de:
  1. Extender el instruction set del microprocesador.
  2. Administrar las interrupciones.
  3. Manejar el reloj de tiempo real.
  4. Conmutar el estado del procesador entre modo supervisor y usuario.
  5. Implementar los mecanismos de protección.

  Sus módulos son:
  1. Controlador de Interrupciones.
  2. Dispatcher (Switcher), que conmuta distintos procesadores y procesos.
  3. Rutinas de Sincronización y Comunicaciones entre Procesos.
  4. Más otras Primitivas.

e) Kernel de un S.O. de TIEMPO REAL:

Las principales características que debe poseer un núcleo para tiempo real son:
1. Atender eventos (interrupciones).
2. Mantener más de una tarea simultáneamente en memoria.
3. Permitir la interrupción de una tarea y reasumir su ejecución en el punto interrumpido.

Sus funciones son :
1. Creación de tareas (bloque de control).
2. Eliminación de tareas.
3. Suspensión / Bloqueo de tareas.
4. Reactivación de tareas.
5. Cambio de prioridades de tareas.
6. Scheduler de tareas.
7. Adelantamiento o retardo de tareas (relacionado con el scheduling).

La creación de tareas requiere en este tipo de núcleos de:
1. Generar nombre.
2. Dar prioridad.
3. Armar el bloque de control.
4. Asignación de los recursos necesarios para esa tarea.

1.9. Prestaciones de un Sistema Operativo. a) Servicios que brinda un Sistemas Operativos. b) Servicios para el usuario. c) Servicios para los programas. d) Servicios para el sistema. e) Protecciones.

Describiremos a las prestaciones de un S.O. desde las dos ópticas: del usuario o programador; y del sistema.

Punto de vista del Usuario o del programador

- **Ejecución de programas**: Para llevar un programa a ejecución el S.O. debe proveer facilidades para cargar un programa en memoria y luego preparar el ambiente como para poder ejecutarlo.
- **Operaciones de entrada** / **salida**: Para lograr esto el S.O. debe proveer las facilidades para que un programa pueda tratar un archivo, enviar o recibir datos a un dispositivo, etc.
- **Manejo de archivos**: las respectivas facilidades de accesos, uso y organización del Sistema de Archivos (File System).

Punto de vista del sistema

- **Asignación de recursos** (Resource Manager): Para ello proveerá un conjunto de Mecanismos de resolución de conflictos en la asignación de recursos cuando varios procesos o usuarios están compitiendo por ellos.
- **Contabilidad** (Account): sobre todo el Control de tiempos de utilización de recursos por los usuarios ya sea para su facturación o simplemente para la obtención de estadísticas de utilización.
- **Protección:** contra acciones no deseadas.

Todas éstas prestaciones las ofrece el S.O. como servicios, protecciones e interfases.

a) Servicios que brinda un Sistemas Operativos

Un sistema operativo provee ciertos servicios a los programas y a los usuarios. Estos servicios difieren de un sistema a otro, pero hay cierta clase de servicios que pueden identificarse como comunes a todos:

- **Ejecución de programas**: El SO debe ser capaz de cargarlos, darles el control y determinar su fin normal o anormal.
- **Operaciones de E/S**: El programa del usuario no puede ejecutar operaciones de E/S directamente, por lo tanto el sistema operativo tiene que proveer ciertos servicios para hacerlo.
- **Manipulación del Sistema de Archivos**: Los programas quieren leer y escribir archivos. Esto se hace por medio del sistema de archivos que dispone el S.O..
- **Detección de errores**: Estos pueden ocurrir en la CPU y memoria, en dispositivos de E/S, o en el programa del usuario. Por cada tipo de error, el sistema operativo tiene que tomar un tipo de acción distinto de acuerdo al error.
- **Administración de recursos**: Cuando hay varios usuarios ejecutando al mismo tiempo, hay que darles recursos a cada uno de ellos. El SO maneja distintos tipos de recursos: CPU, memoria central, archivos, dispositivos.
- **Accounting**: Se quiere contabilizar qué recursos son usados por cada usuario. Estos datos, usados con fines estadísticos para configurar el sistema, pueden mejorar los servicios de computación. También se utilizan estos datos para cobrar a cada usuario los recursos utilizados.
- **Protección**: Los dueños de la información necesitan controlar el acceso y su uso. Cuando se ejecutan distintos trabajos simultáneamente, uno no debe poder interferir con los otros. Las demandas conflictivas para determinados recursos, deben ser administradas razonablemente.

Los servicios de los sistemas operativos son provistos de distintas formas. Dos métodos básicos son las Llamadas al Sistema y los Programas del Sistema. El nivel más fundamental de los servicios se

maneja por medio del uso de llamadas al supervisor. Estas proveen la interfase entre un programa en ejecución y el sistema operativo. Generalmente están disponibles como instrucciones del lenguaje ensamblador.

Podemos reconocer, básicamente, los siguientes tipos de llamadas al kernel:

- Control de Procesos
- Fin, Vuelco (Dump).
- Carga de otro programa (Load).
- Crear procesos, Terminar proceso.
- Esperar un tiempo.
- Esperar por un evento o una señal.
- Crear, borrar archivos.
- Apertura y cierre de archivos.
- Lectura y escritura de Archivo.
- Manipulación de Dispositivos
- Pedir un dispositivo, liberar un dispositivo.
- Leer, escribir.
- Mantenimiento de Información
- Fecha, Hora.
- Atributos de Procesos, Archivos, o Dispositivos

Hay distintas formas de implementar las llamadas al supervisor dependiendo de la computadora en uso. Puede hacerse como interrupciones o como llamadas a un servidor (server). Es necesario identificar la llamada al sistema que se quiere hacer, y otro tipo de informaciones. Por ejemplo, para leer un registro, hace falta especificar el dispositivo o archivo a usar, y la dirección y longitud de memoria donde copiar el registro.

Hay dos métodos generales para pasar parámetros al sistema operativo, la aproximación más simple es pasar los parámetros en registros. Sin embargo, en algunos casos hay más parámetros que registros. En estos casos, los parámetros se almacenan en un bloque o una tabla en la memoria, y se pasa la dirección de ese bloque.

Otro aspecto de un sistema moderno es una colección de programas del sistema. Además del código del kernel residente, la mayoría de los sistemas proveen una gran cantidad de programas del sistema para resolver problemas comunes y otorgar un entorno más conveniente para la construcción y ejecución del programa.

Los programas del sistema se dividen en varias categorías:

- **File Manipulation***:* Estos programas Crean, Borran, Copian, Renombran, Imprimen, Vuelcan, Listan y, generalmente realizan operaciones en Archivos y Directorios facilitando sus usos y manipulaciones.
- **File Modification***:* Uso de Editores de texto para crear o modificar Contenidos de Disco o Cinta. Ejemplo: EDIT, EDLINE en un entorno DOS o el vi para UNIX.
- **Status Information***:* Sirven para dar respuesta a Fecha, Hora, Cantidad de Memoria disponible o en Espacio de Disco, Cantidad de Usuarios, u otra información de estado. La información se formatea y se transfiere o a la Pantalla o a la Impresora.
- **Programming Language Suport***:* Compiladores, Ensambladores e Intérprete para lenguajes de alto *n*ivel, que son a veces provistos por el S.O. (c, qbasic, etc.).
- **Program Loading and Execution***:* El S.O. provee a veces Cargadores absolutos o reubicables, Editores de Enlaces, Cargadores de Overlays, Debuggers, Assemblers, etc.
- **Application Programs***:* A veces los S.O. acompañan programas para resolver problemas particulares, como : Compilador - compilador, Formateador de textos o de discos, Paquetes Gráficos, Sistemas de Bases de Datos, Paquete de análisis estadísticos, etc.
- **Command Interpreter o Shell***:* Algunos S.O. reúnen un conjunto de programas especiales que proveen servicios en un sólo intérprete. Por Ej., MS - DOS.
- **Systems Calls:** Generalmente se invocan cuando se produce una intervención manual (por teclado u otro dispositivo), que produce una interrupción en la secuencia de ejecución.

La Máquina es compartida por los Usuarios, los Programas que se ejecutan y el Sistema Operativo que, además, es un usuario privilegiado, ya que es el árbitro de los conflictos durante la ejecución.

Todos necesitan y requieren servicios para su ejecución, para ello usan las facilidades que provee tanto el S.O., como el Hardware. Todos deben COMPARTIR los escasos Recursos. El S.O. debe velar por su integridad y ser el árbitro en los conflictos, para ello debe proveer adecuadas

protecciones conjuntamente con los servicios.

Describiremos brevemente cada uno de los requerimientos y los respectivos servicios que se devuelven.

b) Servicios para el usuario.

El Usuario requiere un conjunto de servicios para:
- Solicitar la ejecución de un trabajo (Process o Job),
- Manipulación de Archivos y Dispositivos.
- Mantenimiento de la Información de los objetos.

El SO ofrece sus servicios a los usuarios de dos formas diferentes: Las **llamadas al sistema** operativo desde un comando o mandato y la ejecución de programas (**System Programs**) propios del sistema invocados por medio de un comando.

Hay dos métodos:
- Los Programas del Sistema o *System Programs* (se invocan según necesidades).
- Las llamadas al Sistema o *System Calls* (están en instrucciones del Lenguaje Ensamblador o sea preparados para el procesador que ejecutará estas instrucciones).

System Programs provistos por el S.O.: Proveen Soluciones a problemas comunes y frecuentes, pero no durante la ejecución de programas, sino que están disponibles **on - line** a demanda. Ofrecen un entorno para la ejecución más cómodo. Generalmente no están residentes en memoria (excepto en el DOS), sino en un soporte de acceso directo, así cuando se los requiere, se invocan y se cargan rápidamente a memoria para su ejecución. La interacción se hace a través del Shell. Un ejemplo es el programa "Format"

c) Servicios para los programas.

Los Programas para que puedan ser ejecutados necesitan de servicios básicos que brinda el S.O. y estos son:
- **Para la ejecución de los procesos**: el Sistema Operativo brinda los servicios de carga del programa en memoria central, la ejecución y la terminación del mismo. En cuanto a las operaciones de Entrada / Salida se proveen de funciones específicas para cada tipo de dispositivo. Un proceso no tiene la capacidad de comunicarse directamente con los dispositivos de entrada/salida, es función del sistema operativo brindar la interfase.
- **Manipulación de Archivos**: El programa lee y escribe en memoria central. El resto lo hace el S.O. (crea, abre, borra, cierra, graba, usa los periféricos, etc.).
- **Detección de errores**: durante la ejecución de los programas pueden ocurrir errores que se detectan ya sea por el Hardware o por el software generándose una Interrupción. El S.O. trata estas interrupciones adecuadamente.
- **Reparto de Recursos**: entre los Usuarios y los Programas.
- **Contabilidad:** Es relevante para Sistemas multiusuarios y multiprogramados llevar la contabilidad del uso de los recursos.
- **Protección**: brindando adecuadas interfases entre Procesos y Usuarios.

Todos estos servicios son llevados a cabo por el S.O. mediante las llamadas al sistema o System Call. En realidad las llamadas al sistema son las interfases entre un programa en ejecución y el SO.

Las llamadas al sistema se pueden agrupar en los siguientes rubros:
- Para el manejo de procesos.
- Para la ejecución de programas.
- Manejo de operaciones de Entrada/Salida.
- Comunicaciones.
- Manejo del Sistema de Archivos
- Detección de Errores.
- Protección.
- Manejo de recursos (Resource Allocation)
- Contabilidades (Accounting)
- Etc

OBSERVACIÓN: Generalmente se necesita pasar información sobre los objetos para la correcta

ejecución del proceso (Registros, parámetros o bloques de parámetros).

Las llamadas al sistema o System Calls en realidad son rutinas de servicios del SO accesibles desde un lenguaje de programación a través de instrucciones, por ejemplo para algunos sistemas operativos se denominan:

- SVC para IBM
- UUO para DEC PDP 10.
- TRAP Instrucción para PDP 11.

Un System Call esta compuesto por un conjunto de instrucciones en lenguaje de máquina y su apariencia es una macroinstrucción. Su ejecución es atómica o sea sin interrupciones, ya que prefijan el modo de ejecución del procesador en Modo Kernel.

Las llamadas al SO son idénticas a las de una función desde el punto de vista del programa llamador. Después de la llamada, se produce una interrupción del proceso que estaba usando la CPU y se ejecuta la rutina o función del S.O. que ha sido invocada, tomando los datos de los parámetros correspondientes. Al completarse el servicio del S.O. se devuelve el control al programa llamador.

A continuación mencionaremos algunas llamadas al sistema o primitivas de un Sistema Operativo, que pretende ser abarcativa en cuanto a funciones y objetivos que realizan y no son particulares a un determinado S.O.

Respecto a un JOB (conjunto de pasos de trabajo como una rutina):

- **create_job()**: crea un trabajo en el sistema.
- **delete_job():** elimina un trabajo en el sistema.

Respecto Control de procesos (PROCESS):

- **end(), abort()**: finalizar: normalmente; abortar: anormalmente
- **load(), execute()**: cargar, ejecutar: intérprete de mandatos ejecuta programa directamente siguiendo indicaciones de, por ejemplo, un mandato de usuario, pulsar el botón del mouse, o mandato de procesamiento por lotes.
- **create_process(), terminate_process()**: Crear o terminar proceso.
- **get_process_attributes(), set_process_attributes()**: obtener o establecer atributos de proceso.
- **wait_for_time()**: esperar un tiempo.
- **wait-event(), signal_event()**: esperar suceso, señalar suceso.
- **allocate_memory() y free_memory()**: asignar y liberar memoria.

Respecto a una tarea (TASK):

- **create_task()**: crea una tarea.
- **Delete_task()**: elimina una tarea del sistema.
- **Suspend_task()** (sleep): suspende o duerme una tarea.
- **resume_task()**: despierta o reasume la ejecución de una tarea.

INTERRUPCIONES: Dependen del hardware que se esté usando. Existen una serie de instrucciones para tratar a las interrupciones. Estas interrupciones están implementadas entre el Kernel y el I/O System:

- **set_interrupt()**: Asigna un nivel de jerarquía para poder interactuar con el S.O..
- **set_priority()**: Asigna prioridades a procesos.
- **reset_interrupt()**: Revierte la acción de set_interrupt.
- **exit_interrupt()**: Restablece el I/O System.
- **signal_interrupt()**: Es una señal generada por un proceso para que sea tratada por el S.O..
- **wait_ interrupt()**: Obliga al proceso que generó la interrupción a esperar hasta que se pueda tratar la instrucción.
- **enable_interrupt()**: Permite un nivel de interrupción externo.
- **disable_interrupt()**: Inhabilita interrupciones.

SEGMENTOS: Es para pedir más memoria a medida que se necesite:

- **create_segment()**: crea un segmento de memoria.
- **delete_segment()**: elimina un segmento.

MAILBOX o Buffer de mensajes: Se usa para facilitar la comunicación entre procesos:

- **create_mailbox()**: crea un buzón o buffer.
- **delete_mailbox()**: elimina un buzón o buffer.
- **send_mailbox()**: envía un mensaje a un buzón o buffer.
- **receive_ mailbox()**: recibe un mensaje a un buzón o buffer.

EL S.O. UNIX implementa los MAILBOX a través de Pipelines o Pipes (Tuberías).

REGIÓN CRITICA: para los recursos compartidos que se usan concurrentemente:

- **create_region()**: crea una región crítica.
- **delete_region()**: elimina una región crítica.
- **accep_control()**: permite el control de una región crítica.
- **receive_ control()**: recibe el control.
- **send_control()**: envía el control.

SINCRONIZACIÓN:

- **create_semaphore():** crea un semáforo.
- **delete_ semaphore():** (elimina un semáforo).

PRIORIDADES:

- **set_priority()**: fija la prioridad (no confundir con la función de igual nombre de interrupciones).
- **get_priorlty()**: obtiene la prioridad.
- **force_prlority()**: fuerza una prioridad.

MANIPULACIÓN DE ARCHIVOS:

- **create_file(), delete_file()**: para crear y eliminar archivos se requiere el nombre del archivo y quizá algunos atributos.
- **open(), close()**: abrir, cerrar.
- **read(), write(), reposition()**: leer, escribir, posicionar.
- **get_file_attributes(), set_file_attributes()**: obtener o establecer atributos de archivos: nombre, tipo de archivo, código de protección, etc..

MANIPULACIÓN DE DISPOSITIVOS:

- **request_device(), release_device()**: solicitar o liberar dispositivo.
- **read(), write(), reposition()** leer, escribir, reposicionar.
- **get_device_attributes(), set_device_attributes()**: obtener o establecer atributos de dispositivos.
- **logically_attach_device(), Logically_detach_device()**: separar o unir dispositivos en forma lógica.

MANTENIMIENTO DE INFORMACIÓN:

- **get_time() o get_date(), set_time() o set_date()**: obtener o establecer hora o fecha.
- **get_system_data(), set_system_data()**: obtener o establecer datos del sistema.
- **get o set: process, file, o device attributes()**: obtener o establecer atributos de proceso, archivo o dispositivo.

Hay llamadas al sistema que existen para transferir información entre el programa de usuario y el S.O. Este último conserva información de sus trabajos y procesos y se cuenta con llamadas al sistema para obtener y restablecer esta información

COMUNICACIÓN:

- **create_communication_connection(), delete_communication_connection()**:crear o eliminar conexión para comunicación.
- **send_messages(), receive_messages()**: enviar a recibir mensajes.
- **transfer_status_information()**: transferir información de estado.
- **Attach(), detach_remote_devices()**: conectar o desconectar dispositivos remotos.

El procesamiento de una llamada al sistema o mecanismo básico de la ejecución de un System Call. La interacción la hace el programa en ejecución con el S.O. sin intervención del usuario produciendo el cambio de contexto.

d) Servicios para el sistema.

Los servicios para si mismo lo hace mediante una función que se invoca y ejecuta completamente. Por ejemplo la inicialización de la máquina o ciertas invocaciones por excepciones que han ocurrido en la ejecución.

- El intérprete de comandos y los programas del sistema son los que fijan el entorno y la forma de ver el SO por los usuarios.
- En cambio, el programador del sistema tiene una visión completamente diferente; para él todo son recursos físicos y dispositivos que deben ser convertidos en entidades lógicas para ofrecérselas a los usuarios.
- Un SO es un programa activado por eventos (event-driven). Normalmente cada evento produce una interrupción de la ejecución del S.O.

Llamadas al SO: ocurren cuando se produce:

- Terminación normal de un programa en ejecución (**end** - el SO realiza una serie de tareas y luego transfiere usando un comando al Shell)
- Terminación anormal: lo hace mediante el tratamiento de errores.
- Peticiones de estado (pedido de Fecha, hora, espacio de Memoria Central, etc). El S.O. computa la solicitud y devuelve el control al Programa en ejecución.
- Peticiones de recursos: El programa necesita más recursos luego el S.O. si dispone le asigna de lo contrario lo demora.
- Peticiones de E/S: cuando se requiere realizar una operación sobre un dispositivo.

Interrupciones de los dispositivos de E/S (I/O Device Interrupt): Una vez que un programa en ejecución realiza una E/S, se pueden tomar dos tipos de acción por parte del SO:

- Proceso queda en espera hasta que finalice la E/S. *(**Wait, o Loop: Jump Loop**,* u otra solución: Una tabla con campo State 'IDLE" "BUSY" etc.). El dispositivo externo producirá una interrupción, al finalizar la operación, que dará el control al S.O. y activará al proceso que estaba suspendido esperando que se complete el servicio de E/S.
- Proceso sigue ejecutando mientras se realiza la E/S. El dispositivo también produce una interrupción al S.O. pero el proceso no es activado sino que se le indica que la operación solicitada finalizó.

Manejo de excepciones (traps): se producen cuando un programa en ejecución comete un error (división por cero, acceso a zonas prohibidas, ejecución de instrucciones privilegiadas, etc.). El S.O. asocia alguna función para el tratamiento de estos errores.

Estos problemas son tratados con ciertos requerimientos de seguridad en la realización de las tareas por lo que interviene el S.O. como un órgano de control para brindar los respectivos servicios.

e) Protecciones

Tienen el objeto de evitar problemas entre los usuarios y los procesos y éstos con el S.O. Evitan tanto los errores de programación como las malas intenciones de un usuario o programa en los accesos a recursos.

Se debe prever las siguientes protecciones:

- **Protección de E/S:** Es llevada a cabo por los drivers de dispositivos que devuelven control al S.O. cuando una situación de error se produce.
- **Protección de memoria:** Cada proceso tiene una zona de memoria asignada para su funcionamiento (espacio de direccionamiento privado). Una de las formas de evitar que el proceso acceda fuera de su espacio de direccionamiento, es usar dentro de la CPU un registro para el limite inferior y otro registro para el límite superior y controlar en cada direccionamiento si el acceso se realiza dentro de estos dos límites para el usuario 1.

- **Protección del procesador:** Otro tipo de problema es la presencia de ciclos infinitos o accesos al procesador por procesos que no lo liberan de su uso nunca. Para evitarlo, el hardware incluye un reloj (clock), que marca períodos de tiempo, de manera que al terminar un período se produce una interrupción y el SO puede tomar control de la CPU.

1.10. SISTEMAS OPERATIVOS PARA MULTIPROCESADORES.

a) Tipos de S.O. para multiprocesadores. b) Funciones de S.O. para Multiprocesadores. c) Sistemas de multiprocesador con tiempo compartido.

a) Tipos de S.O. para multiprocesadores:

Existen los siguientes tipos básicos:
1. Maestro - esclavo
2. Supervisores separados.
3. Simétricos (es el más común).
4. Sistemas de multiprocesador con tiempo compartido

1. Maestro - Esclavo (MASTER - SLAVE): Posee estructura jerárquica, tradicional, pero requiere que todos los procesadores sean idénticos (homogéneos).

MASTER:
- Planifica los trabajos. Cuando un S.O. usa el procesador y lo hace maestro, planifica para todos en ese momento. Se maneja con un sistema de mensajes (no hay interrupciones)
- Controla la ejecución de los esclavos.
- El Sistema Operativo puede ejecutarse en cualquier procesador (maestro flotante)
- Puede tomarse a cualquier procesador como master (con este método se manejan la arquitectura de procesadores llamada hipercubos).

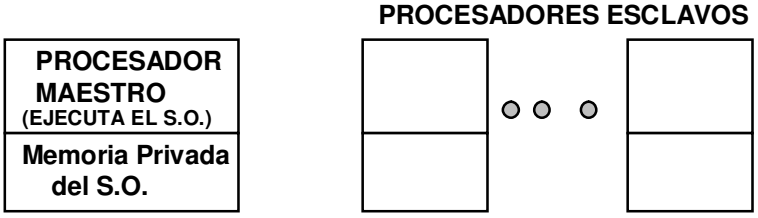


Figura 1.46 Procesadores maestros - esclavos.

VENTAJAS:
- Sencillo de implementar y fácil de adaptar.
- Permite paralelismo en las aplicaciones pero no en el S.O.

DESVENTAJAS:
- Los esclavos tienen como cuello de botella al master.
- Gran overhead[9] por la estructura jerárquica.

1. **Supervisor separado**:
   - Cada nodo tiene su propio kernel que gestiona el procesamiento del nodo (procesador, Memoria, E/S), en forma independiente (caso del hipercubo o crosstalk).
   - Cómo lograr el paralelismo de la aplicación?
     - Subdividiendo la aplicación en tareas (task), del sistema.
     - Cada Nodo ejecutará una tarea.

[9] Overhead: es la cantidad de tiempo empleado en la ejecución del S.O. para brindar servicios a los programas en ejecución. Es un tiempo no productivo.

- La planificación macroscópica de la ejecución de las aplicaciones lo hace un SUPERVISOR que puede implementarse en modo simétrico o Master-Slave
- Las funciones del Supervisor son: Asignación de procesadores a tareas, Mensajes, Redireccionamientos, Almacenamientos, etc..

2. **Simétricos** / **Asimétricos**:

a) **Simétricos**:

- Todos los procesadores son funcionalmente idénticos. Cualquier procesador puede acceder a cualquier dispositivo de I/O o a cualquier Memoria, menú, etc. Por esto se dice que es simétrico.
- El Sistema Operativo puede ser ejecutado en cualquier procesador e incluso en paralelo en varios procesadores. Para ello deben diseñarse y construirse funciones que ejecutan autónomamente y deben existir adecuados controles y sincronizaciones para las estructuras comunes o compartidas.
- La implementación más sencilla es la de "Maestro Flotante".
- El S.O. no está ligado a ningún procesador y flota de uno a otro. Por lo general, el código del S.O. es monolítico por lo que se ejecuta en un solo procesador. El S.O. es una sola "Región Critica[10]" de grandes dimensiones. Temporalmente el procesador que ejecuta el S.O. es el Maestro y planifica el trabajo para los demás procesadores.

Ejemplo UNIX: La ejecución es seleccionada de la cola de procesos listos en Memoria Compartida, la asignación se hace:

- Primer proceso a primer procesador y así sucesivamente hasta agotar procesadores o procesos.
- El Procesador que se libera toma el siguiente proceso de la cola de listos para ejecutar.
- La concurrencia se administra mediante funciones que controlan las interrupciones (habilitaciones/deshabilitaciones).
- No funcionan las prioridades en multiprocesadores (es un mecanismo de monoprocesador).
- Funcionan los hilos (threads)

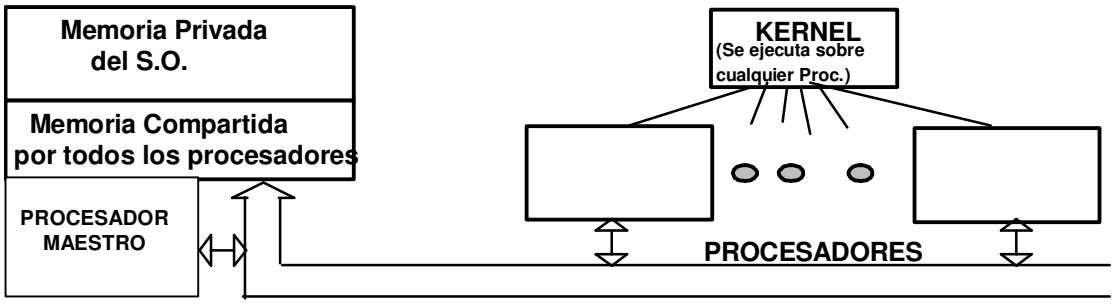


Figura 1.47 Procesamiento Flotante.

VENTAJAS:

- Fácil de portar de S.O. monoprocesador a multiprocesadores
- Se logra buena ejecución en paralelo.

DESVENTAJA:

- Difícil reprogramación del control de la concurrencia.

b) **Asimétricos**:
Solo algunos procesadores tienen acceso a Memoria y dispositivos

b) Funciones de S.O. para Multiprocesadores:

[10] Región crítica: solo un proceso por vez ejecuta en ella

- Gestión eficiente de recursos (procesadores, memoria central, E/S).
- Planificación de procesadores (multiples). Esto significa dos cosas:
  - § Asignación de procesadores a las aplicaciones en forma consistente.
  - § Asegurar el uso eficiente de los procesadores asignados a las aplicaciones.
    - Esto afecta a la velocidad.
    - La velocidad y la productividad dependen de las prioridades - o sea de la planificación.
- Mecanismos y facilidades que fomentan el paralelismo.
  - Mecanismos de sincronización flexibles y eficientes entre procesadores y entre procesos.
  - Creación y gestión de un gran número de threads.
  - Gestión de Memoria Central: depende de la Arquitectura y del interconexionado (fuertemente o débilmente acoplados).
  - Gestión de dispositivos de E/S (Generalmente se usan arrays de discos RAID con buses SCSI).

c) Sistemas de multiprocesador con tiempo compartido

Los **Sistemas de multiprocesador con tiempo compartido**: Ofrecen la imagen de un único sistema pero lo hacen mediante la vía de centralizar todo, por lo que en realidad este caso es un único sistema. Los multiprocesadores con tiempo compartido *NO son sistemas distribuidos*.

Este tipo de sistemas es una combinación de software fuertemente acoplado en un hardware fuertemente acoplado. Aunque existen varias máquinas de propósito especial en esta categoría, los ejemplos más comunes de propósito general son los multiprocesadores, que operan como un sistema de tiempo compartido de UNIX, solo que con varios procesadores idénticas en vez de uno solo. Para el mundo exterior, un multiprocesador con 32 procesadores idénticos de 10 MIPS actúa de manera muy parecida a un solo procesador de 320 MIPS. Esta es la imagen de un único sistema analizada antes. Solo que la implementación de éste en un multiprocesador es más sencillo, puesto que todo el diseño se puede centralizar. Este esquema es muy utilizado en equipos servidores escalables y su configuración se llama Clusters.

La característica de este tipo de sistemas es la existencia de una sola cola de ejecución: una lista de todos los procesos en el sistema que no están bloqueados en forma lógica y listos para su ejecución. La cola de ejecuciones una estructura de datos contenida en la memoria compartida.

Como por ejemplo considere la figura 1.48, con 4 procesadores y 6 procesos listos para su ejecución. Los seis procesos están dentro de la memoria compartida y por el momento se ejecutan 4 de ellos: el proceso a en el procesador P 1, el proceso B en el P 2 , el C en el P 3 y el proceso D en el P 4. Los procesos E y F están en la memoria esperando para ser ejecutados.

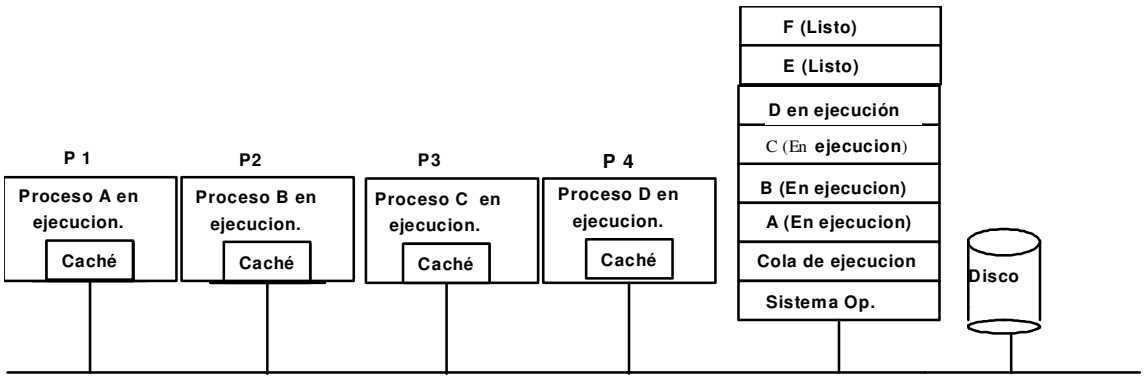


Fig. 1.48. Un sistema multiprocesador con una sola cola de ejecución.

Supongamos que el proceso B se bloquea en espera de entrada/salida o que su quantum de tiempo haya terminado. Es el P 2 quien debe suspenderlo y encontrar otro proceso para llevarlo a ejecución. Lo normal sería que el P 2 comenzará la ejecución del dispatcher del sistema

operativo (localizado en la memoria compartida). Después de realizar el cambio de contexto (context switch) de la ejecución y guardarlo en memoria central, entrará a una región critica para ejecutar sobre B el planificador y buscar otro proceso para su ejecución. Es esencial que el planificador se ejecute como una región critica, con el fin de evitar que dos procesadores seleccionen el mismo proceso para su ejecución inmediata.

Puesto que ninguna de los procesadores tiene memoria local y todos los programas se almacenan en la memoria global compartida, no importa sobre que procesador se ejecute un proceso. Si un proceso con un tiempo de ejecución grande se planifica muchas veces antes de terminar, en promedio, gastará la misma cantidad de tiempo que si se ejecutase en un procesador una sola vez.

Un aspecto colateral es que si un proceso se bloquea en espera de E/S en un multiprocesador, el sistema operativo tiene la opción de suspenderlo o bien dejarlo que realice una espera ocupada. Si la mayoría de la E/S se lleva a cabo en menos tiempo del que tarda un cambio entre los procesos, entonces es preferible una espera ocupada. Algunos sistemas dejan que el proceso conserve su procesador por unos cuantos milisegundos, con la esperanza de que la E/S termine pronto, pero si esto no ocurre antes de que se agote el tiempo, se realiza una conmutación de procesos sobre ese procesador.

Un área en donde este tipo de multiprocesador (Clusters) difiere de manera apreciable de una red o un sistema operativo distribuido es la organización del sistema de archivos. Cuando cualquier proceso ejecuta una llamada al sistema operativo, éste se ejecuta mediante un sistema de sincronización ya sea semáforos, monitores, o cualquier otro equivalente para bloquear los demás procesadores, mientras se ejecutan las regiones criticas o se tiene acceso a las tablas del Sistema. De este modo, por ejemplo al hacer una llama WRITE, el caché central se bloquea, los nuevos datos entran al caché y luego se retira el bloqueo.

Se deja expreso, que los métodos utilizados en el multriprocesador para lograr la apariencia de un uniprocesador virtual no se pueden aplicar a máquinas sin memoria compartida. Las colas centralizadas de ejecución y los cachés de bloque solo funcionan cuando todas las CPUs tienen acceso a ellas con muy poco retraso. Aunque estas estructuras de datos se podrían simular en una red de máquinas, los costos de comunicación hacen que este método sea prohibitivamente caro.

1.11. Bibliografía recomendada para el módulo 1.

Operating Systems. (Fith Edition), Internals and Design Principles, Stallings Willams, Pearson-Prentice Hall, Englewood Cliff, NJ., 2005,

Applied Operating Systems Concepts Fiftth Edition, Silberschatz, A.; Galvin P. B and Gagne G., Wiley, 2003,

Operating Systems Concepts and Design (Second Edition), Milenkovic, Millan., Mc Graw Hill, 1992

Modern Operating Systems, (Secund edition) Tanenbaum, Andrew S., Prentice - Hall International, 2001

Operating Systems, Design and Implementation (Second Edition), Tanenbaum, Andrew S., Prentice - Hall International, 1997

Fundamentals of Operating Systems., Lister, A.M. , Macmillan, London; 1979

Operating System Design - The XINU Approach; Comer, Douglas; Prentice may; 1984

Operating System; Lorin, H., Deitel, H.M.; Addison Wesley; 1981

The UNIX Operating System; Christian, K.; John Wiley; 1983

UNIX System Architecture; Andleigh, Prabhat K..; Prentice - Hall International; 1990

The UNIX Programming Environment; Kernighan, B.W. and Pike, R.; Prentice - Hall International; 1984

The UNIX System V Environment; Bourne, Stephen R.; Addison Wesley; 1987

Sistemas Operativos Modernos (Segunda Edición) Tanenbaum A. S.; Pearson – 2003
Fundamentos de Sistemas Operativos (Séptima Edición), Silberschatz, A.; Galvin P. B and Gagne G., Mc Graw Hill, 2005

Sistemas Operativos Conceptos y diseños.(Segunda Edición); Milenkovic Milan.; Mc Graw Hill; 1994..

Sistemas de Explotación de Computadores; CROCUS; Paraninfo; 1987 424 pág.

GLOSARIO DE TÉRMINOS EN IDIOMA INGLÉS

Mainframe	Host	Bootstrap	Bootstrapping
Workstation	Personal Computer	Multiaccess	Remote Job Entry
Batch	Analogic Digital Converter	Digital Analogic Converter	Fault tolerance Operating System
Fault tolerance System	Feedback	On-line	Off-line
I/O System	I/O Interrupt	Networking	Release
I/O Extensions	Kernel	Shell	Job
I/O Device Interrupt	Job Control Language	Loader	Dispatcher
I/O Ports	Communication Manager	Process	Warm Start
Prompt	User	Cold Start	System Calls
Supervisor Call	Atomic Action	User Friendly	Super User
Command Line Interpreter	Command Line interface	System Manager	write
Binding	Delay Binding Time	Assign	Information hiding
Rings	Trap	Time Sharing	Botton Up
Microkernel Architecture	Message	Top Down	Spoolers
Set	Instruction set	Message passing	Switcher
Switches	Run Time	Run Time Environment	Halt
Exit to User	Program Counter	Interrupt Request	Interrupt Acknowledge
First Level Interrupt Handler	Polling	Interrupt Vector System	Mask
Interrupt Service Routine	Program Check	Page Fault	Invalid Instruction
Overflow	Underflow	Safe Point	Direct Memory Access
Resource	Resource Manager	Resource Allocation	Account
Program Calls	System Program	Status	Status Information
Programming Language Support	Program Loading and Execution	Application Programs	Command Interpreter
Overlay	Delete	End	Abort
Load	Execute	Terminate	Get
Attribute	Set	Wait for time	Wait for event
Signal	Allocate Memory	Free	Suspend task
Sleep task	Resume Task	Set priority	Reset Priority
Enable	Disable	Mailbox	Send
Receive	Accept Control	Semaphore	Open
Close	Read	Write	Reposition
Jump Loop	Loop	Device	Logically Attach device
Logically Detach device	Time	Date	Transfer
Event Driven	Idle	Busy	Process Control Block
Treads	Server	Initiate	Kill
New	Ready	Queue	Leader
Running	Waiting	Job Queue	Completed
Process Identifier	Tree	Ready Queue	Graphical user interface
Query	Return Value	Orgware	Firmware
Fork	Deadlock	Hardware	Software
Shell	Caller	Peopleware	

GLOSARIO DE TÉRMINOS EN CASTELLANO

Sistema
Booteo
Sistema de Consulta de Información
Sistema de Propósitos especiales
Control de Procesos
Kernel
Recurso Crítico
Inicialización
Máquina Extendida
Política
Protección
Reparto de recursos
Ejecución dual de instrucciones
Estructura de un S.O.
Minidisco
Vuelco de Memoria
Modo Kernel
Bloque de Control del Proceso
Gestor de Interrupciones de 1er nivel
Llamada al Sistema
Manejo de excepciones
Ejecución atómica
Operativo
Monousuario
Sistema de Gestión de Operaciones
Entrada Remota de Trabajo
Componentes de un S.O.
Recurso Computacional
Recurso común
Primitivas
Configurar el Sistema
Estrategia
Contabilidad
Gestión de la información de objetos
Mutua Exclusión
Arquitectura de un S.O.
Red de Máquinas Virtuales
Instrucciones del Usuario
Modo Usuario
Interrupción
Rutina de atención de interrupción
Servicios del S.O.
Contexto de Ejecución
Sistema Operativo
Multiusuario
Sistema de Propósitos Generales
Entrada en Tiempo Real
Recurso Compartido
Funciones de un S.O.
Amigable
Arbitro imparcial
Autoridad
Características comunes de los S.O.
Cooperación entre procesos
Ocultamiento de la información
Monitor Virtual
Juego de Instrucciones
Instrucciones Privilegiadas
Vector de Estado
Pedido de atención de Interrupción
Enmascaramiento de una Interrupción
Programas del Sistema
Computación

ACRÓNIMOS USADOS EN ESTE MÓDULO

S.O. / SO	Sistema Operativo
I.P.L.	Initial Program Loader
P.C.	Personal Computer
R.J.E.	Remote Job Entry
E/S	Entrada / Salida
I/O	Input / Output
ADC	Analogic Digital Converter
DCA	Digital Analogic Converter
ROM	Read Only Memory
RAM	Random Access Memory
J.C.L.	Job Control Language
HW	Hardware
P.C.	Personal Computer
CLI	Command Line Interpreter
RI	Registro de Instrucciones
PC	Program Counter
CPU	Unidad Central de Procesamiento
GUI	Graphical User Interface
M.C.	Memoria Central
THE	Technical Hoeschcool Eindhover
V.M.	Virtual Machine
MV	Máquina Virtual
USR	User
IRQ	Interrupt Request
INTA	Interrupt Acknowledge
FLIH	First Level Interrupt Handler
RAI	Rutina de atención de Interrupción
PAI	Pedido de Atención de interrupción
SW	Software
DMA	Direct Memory Access
DRAM	Dinamic RAM
DEC	Digital Equipment Corporation
LRU	Least Recently Used

Anexo 1.A: Interrupciones.

Introducción a las Interrupciones

Definimos como **interrupción** a la detención del programa en ejecución debido a que el procesador es forzado a reconocer mediante una señal, que un evento ha sucedido en el sistema. En otras palabras, **interrupción** es la detención del programa en ejecución debido a que el procesador es forzado a reconocer mediante una señal generada por un evento que ha ocurrido en el sistema.

En general, para el tratamiento de cualquier interrupción, el procesador debe preservar los contenidos de todos sus registros y cierta información acerca del estado del proceso (salva en el stack de la memoria central, el VECTOR DE ESTADO o Bloque de Control del Proceso -PCB-), reemplaza el contenido del contador de programa (Program Counter - PC), por una dirección predeterminada, que lee desde el bus de datos y que es generada por el Procesador de interrupciones (First Level Interrupt Handler) y esa dirección apunta al comienzo del área de Memoria Central donde se encuentra la rutina de atención de interrupciones. A continuación, el procesador comienza a ejecutar la rutina de interrupción, que es un programa escrito para responder a la condición generada por el pedido de atención de esa interrupción (PAI-IRQ).

Cuando se habla de multiprogramación, es porque hay varios procesos que se encuentran en estado "activo" (no de "ejecución"), ya que han sido comenzados pero aun no han sido completados o terminados (pero solo pueden utilizar el procesador de a uno a la vez). La máquina almacena estos programas en la memoria central.

Clasificación de las interrupciones:

Las interrupciones pueden ser clasificadas según dos criterios distintos.

El primero de los criterios se basa en la prioridad que tendrá la interrupción.

El segundo criterio en el origen de la misma.

- **Clasificación según su prioridad:** Las interrupciones se clasifican como no enmascarables, y enmascarables.
  - **No Enmascarables:** Son las interrupciones de mayor prioridad en el sistema. Cuado se detecta una interrupción de este tipo, el sistema operativo detiene lo que esta haciendo y se ocupa de atender la interrupción. Éste tipo de interrupción puede detener hasta la ejecución de instrucciones en modo Kernel.
  - **Enmascarables**: Son interrupciones de menor prioridad (dentro de las interrupciones, pero siempre tienen mayor prioridad que cualquier proceso usuario). Cuando se produce una de éstas interrupciones el sistema operativo primero permite que se termine de ejecutar la instrucción en curso, y luego atiende la interrupción (se podría decir que la interrupción es encolada, y atendida luego). Éste tipo de interrupciones puede ser interrumpida (valga la redundancia), por una interrupción de mayor prioridad (generalmente del tipo anterior).
- **Clasificación según su origen:** Las interrupciones se clasifican según quien es el causanante de la interrupción. Estas se dividen en originadas por software y las causadas por el hardware. Dentro de éstas últimas se subdividen en internas y externas.
  - **Software:** Son interrupciones generadas por los programas. Dentro de ésta categoría se encuentran todos los Systems Calls.
  - **Hardware:** Son interrupciones generadas por algún componente físico del sistema de computación. Se subdividen en dos subcategorías:
    - **Internas**: Se producen dentro del procesador (internas al procesador). El ejemplo más importante de éste tipo de interrupciones es la división por cero, otro ejemplo es el código de operación inválido, etc..
    - **Externas**: Son interrupciones que se producen fuera del entorno del procesador (externas al procesador). Están relacionadas con los

dispositivos periféricos, como ser por ejemplo finalización de una operación de entrada / salida, falla, defecto o error en algún dispositivo de hardware.

Independientemente del origen de la interrupción, hay un tratamiento que lo realiza el hardware y un tratamiento que lo hace el software.

a) Tratamiento de las interrupciones por el hardware:

Desde el punto de vista del procesador, una interrupción no es más que un bit que se enciende en un determinado sector del procesador denominado FLIH (First Level Interrupt Handler).

Si un programa fuera el único usuario de la CPU, ésta estaría ociosa mientras el programa está esperando la terminación de sus operaciones de E/S, las cuales insume tiempos que generalmente degrada el rendimiento del sistema. Para evitar desperdiciar un recurso tan valioso como lo es el tiempo en que la CPU está parada, se ha propuesto a que haya varios programas activos simultáneamente. A esto lo definiremos como multiprogramación. Los programas en un sistema multiprogramado son independientes entre sí (no siempre, ni necesariamente como se verá más adelante), y residentes en Memoria Central para alternarse en la ejecución sobre una sola CPU en forma entrelazados en el tiempo.

Los programas se encuentran activos o en estado de ejecución cuando tienen asignados, por el Sistema Operativo, todos los recursos que necesitan para poder ser ejecutados (menos el procesador). Residen en Memoria Central, y ya fue ordenada su corrida, aunque no dispongan aun del procesador.

No debe confundirse multiprogramación con multiprocesamiento que significa más de un procesador trabajando sobre la misma memoria central.

En multiprogramación hay tres modos para interrumpir la ejecución de un programa:

- **SIN INTERRUPCIÓN DE PROGRAMA**: El programa conserva el control hasta que termina o hasta que necesita efectuar una operación de E/S. Luego recupera el control el Sistema Operativo quien decide a que programa de la cola de espera, le cede nuevamente el control. La mayor desventaja de esta solución es que un programa de baja prioridad puede conservar el control tanto tiempo como lo desee, mientras no se interrumpe por una E/S o termine. Esta forma de ejecutar se conoce como por lote o batch.
- **CON INTERRUPCIÓN DE PROGRAMA**: La diferencia con el sistema anterior es que, el Sistema Operativo recupera el control a través de un mecanismo de interrupciones. En ese momento busca el programa de mayor prioridad, entre los programas que no están bloqueados (esperando que finalice una operación de E/S), para cederle el control.
- **CON INTERRUPCIÓN POR TIEMPO**: Todos los programas comparten equitativamente el tiempo de uso de la CPU (Hay distintas variantes para administrar el tiempo). Existe un tiempo máximo. Para marcar el tiempo se utiliza un mecanismo de relojería que produce interrupciones a intervalos de tiempo constantes. Además un programa puede perder el control de la CPU cuando se bloquea por necesidades de E/S.

Las interrupciones permiten detener provisoriamente el desarrollo o ejecución de un programa en curso para ceder la CPU a otro programa de mayor prioridad. Cuando esto ocurre, toma el control de la CPU un módulo del Sistema Operativo encargado del tratamiento de las interrupciones (Parte del Kernel). Antes de entregarle el control de la CPU a este modulo, un proceso del Kernel se ocupa de preservar el estado de la máquina, a fin de poder reanudar posteriormente la ejecución del programa interrumpido. El estado de la máquina son todos los contenidos de los registros que, luego se almacenan en la pila de ejecución.

Recordemos entonces que una interrupción es una señal o mecanismo que fuerza a reconocer la ocurrencia de un evento en el sistema.

Para comprender el funcionamiento del mecanismo de interrupciones propondremos un procesador genérico al cual se le pueden hacer extensiones o adaptaciones para cualquier modelo de computador.

Normalmente entre el procesador y los dispositivos periféricos ocurren una serie de señales de avisos o solicitudes, o pedidos de interrupción, las que deben ser atendidas o tratadas, por la CPU o algún programa, lo más rápidamente posible en su reacción.

La procedencia y significado de la Señal pueden ser de diversas índoles. A continuación ejemplificaremos algunos eventos que producen interrupciones. Las causas de interrupción pueden ser:

- Por **fallas del hardware:** falla de un módulo, falta de inicialización, dispositivo desconectado, etc.
- Por **causa de programa**: direccionamiento a un área de memoria que no le corresponde, ejecución de instrucciones privilegiadas (del Sistema Operativo), por el usuario, instrucciones de interrupción de secuencia, etc.
- **Por E/S:** generada por el canal o controlador cuando finaliza una operación de E/S, o por error producida en ésta.
- **Por causa externa:** por intervención del operador, por falla de alimentación eléctrica, por otro computador, etc.
- Por el dispositivo contador de tiempo (**Reloj**), usado en tiempo compartido (time sharing), para administrar el uso compartido del procesador.

En todo el Sistema (Módulos internos, como ser: Memoria Central, Reloj, Unidad de Control, canales de E/S, Controlador de Interrupciones, etc., y los módulos externos: Convertidores Analógicos-Digitales –DAC-, y Convertidores Digitales-Analógicos –ADC-, Sensores, Dispositivos de Prefijación de valores, Impresoras, Floppy-Disk, Consola de vídeo o pantallas, etc., con sus respectivos controladores), ocurren señales por distintos motivaciones que llamaremos "**pedido de atención de interrupción** (PAI)" (interrup request o irq).

EJEMPLO 1: En el propio microprocesador, o en caso que se disponga de un Co-procesador matemático (APU), acoplado, puede ocurrir que se produzca una división por cero, lo que ocasionaría un Overflow (Desborde), en el registro correspondiente al acumulador del procesador. El pedido de interrupción se produce en el bit de Overflow del Registro de FLAGs de la CPU. En el APU existe una palabra de estado (***status word***), en el que un bit, o una combinación de bits, indica (cambia de estado), cuando ocurre una división por cero, cuyo valor puede ser consultado por un programa a través de un canal de entrada.

EJEMPLO 2: En una unidad de Floppy-Disk o disquetera se verifica, mediante un sensor lumínico o magnético, constantemente la posición del agujero "Índice" del disquete, cuya ubicación determina el "***sector cero***". Cuando se detecta este agujero el controlador de la unidad de Flopy-disk genera una señal cuyo significado es: "Comienza el ***sector cero***".

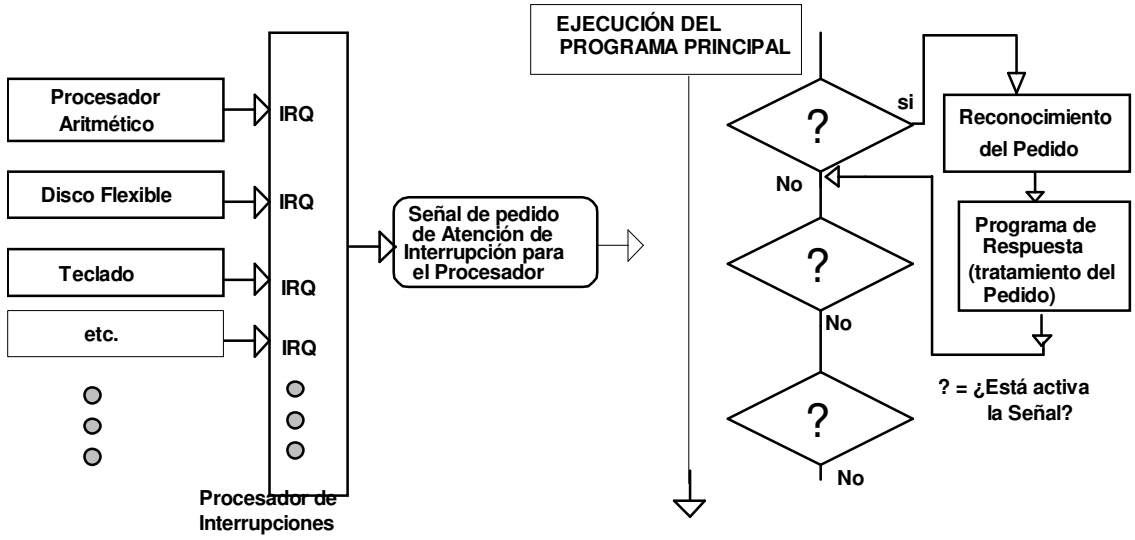


Fig. 1.A.1. Interrupciones por polling

EJEMPLO 3: En el caso de que en un computador se a sobrepasado un valor prefijado, inmediatamente se genera un aviso (señal), que indica que se ha alcanzado o sobrepasado o que mediante la resta se ha obtenido el valor cero del valor previamente prefijado.

EJEMPLO 4: En un canal o Puerta de Entrada/Salida (I/O PORTs), por el cual se están introduciendo datos, se ha generado una señal que dice "***input buffer full"*** (IBF), o sea en la memoria donde se acumulan los datos ha sido llenada por ese periférico. Esta señal IBF se presenta como un uno ("1" binario), lógico ya sea en un pin de un chip o en un cable del bus de control, lo que se transmite a través de una palabra de estado, la cual puede ser consultada o interpretada por un programa.

Clasificación de las ocurrencias que generan Pedidos de Interrupción

Los pedidos de interrupción se pueden resolver mediante un tratamiento llevado a cabo por dos métodos: por SOFTWARE o por HARDWARE a través de una lógica de tratamiento adecuada a cada pedido en particular que depende del evento causante del mismo. Por ello se puede clasificarse en tres grandes grupos a saber:

- **Instante de la ocurrencia**: en
  - ***Ocurrencia esporádica***: por ejemplo: Se ha sobrepasado el valor prefijado, etc..
  - ***Ocurrencia regular o periódica***: por ejemplo: Comienza el sector CERO.
- **Frecuencia de ocurrencia** (Oportunidad o Exuberancia): en
  - ***ocurrencia frecuente***: *por ejemplo: Se ha llenado el Buffer IBF en el caso de que el canal de Entrada sea muy utilizado, etc..*
  - ***ocurrencia rara o poco frecuente***: *por ejemplo: la división por cero.*
- **Importancia del pedido o prioridad**: en
  - ***importante o muy importante***: por ejemplo: en el caso de la división por cero, lo que debe ser tratado inmediatamente desde el programa con una reacción enérgica dado que desde allí se puede arrastrar el error a otros resultados.
  - ***ocurrencia poco importante***: por ejemplo: un pedido de menor importancia en el caso donde no se requiera una reacción inmediata.

Tipos de lógica de interrogación

Depende, si los pedidos tienen alta prioridad o son frecuentes en la ocurrencia o son esporádicos, el criterio que se use para la interrogación de las líneas que conducen las señales de Pedido y su frecuencia.

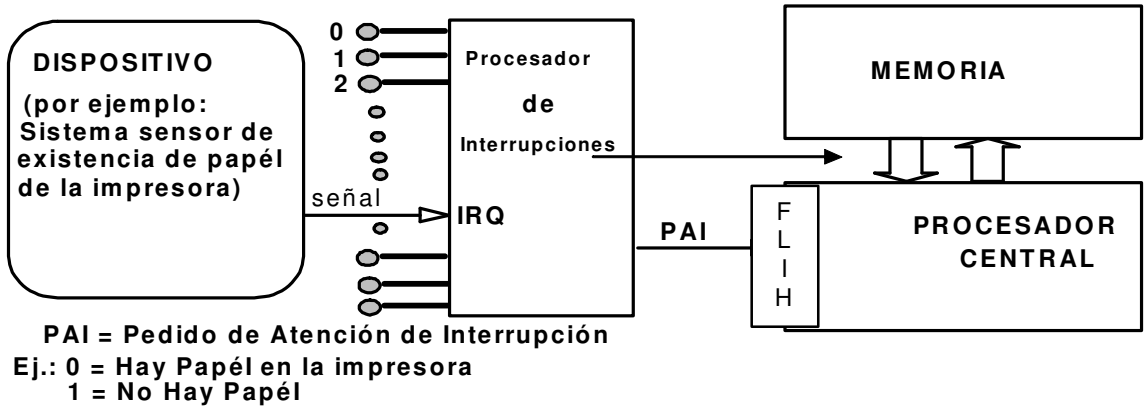


Fig. 1.A.2. Tratamiento de una interrupción

Si la prioridad es alta, además de la interrogación frecuente, la reacción de la respuesta debe ser lo más inmediata posible, lo que debe ser resuelto mediante un Sistema de Interrupción (Interrupt - System), realizado por Hardware. En este caso las interrogaciones al proceso ocurren al final de cada instrucción. Con ello se logra alta seguridad (garantía), en el rápido tratamiento.

Si los pedidos IRQ (solicitudes o avisos), no son de alta prioridad y tampoco muy frecuentes en su ocurrencia, o como en el caso de los programas (en los cuales queda predeterminada), la interrogación se puede realizar por Software en un canal de Entrada. Entonces, la pregunta (Interrogación del estado), se efectúa periódicamente por programa, como un contenido propio del mismo y a un predeterminado instante. Esta técnica es más simple y económica que la técnica del sistema de interrupción por hardware, y se denomina Operación de interrogación por software o POLLING (Escrutinio o interrogación).

La lógica de respuesta observa, ante las dos técnicas: sistema de interrupción ó de Polling, un comportamiento semejante: después del reconocimiento del pedido, interrumpe el procesamiento del programa que se estaba ejecutando en el instante que apareció el Pedido IRQ, salva el estado de la máquina y produce un salto a un programa de tratamiento de la interrupción.

Después de la corrida de éste programa de "tratamiento de la interrupción (el que lleva a una atención del periférico solicitante del pedido y su tratamiento por la CPU), sigue en proceso el programa principal en el lugar (punto), en que se produjo la interrupción. Esto se puede ver en la figura siguiente.

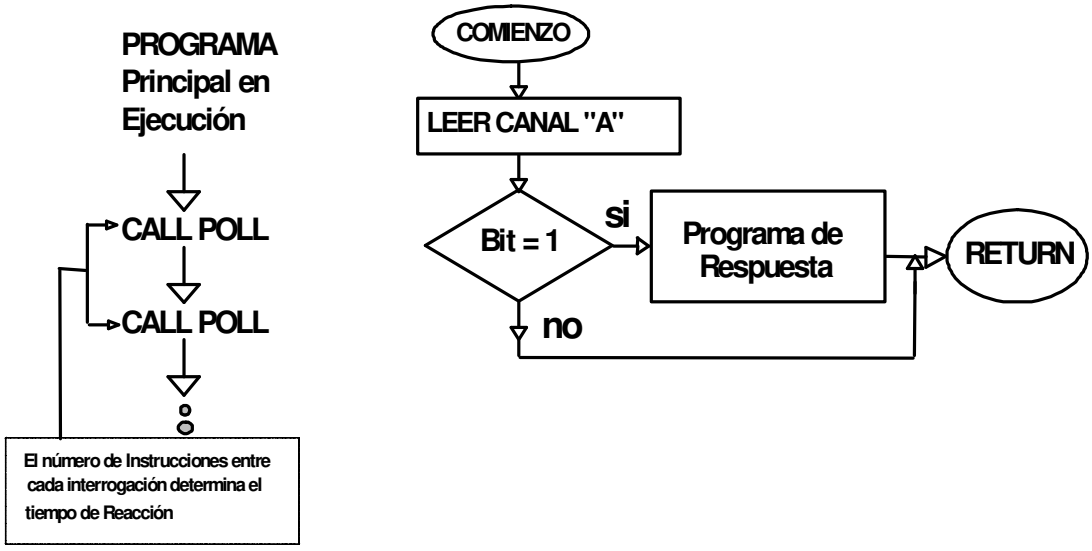


Fig. 1.A.3. Mecanismo del tratamiento de interrupción por polling

Para la interrogación y su comparación del Pedido de Interrupción, por ejemplo: "falta de papel en la impresora", sería excelente la elección de la técnica del polling. En este caso se puede elegir un periodo cíclico, de por ejemplo 2 milisegundos o 2000 instrucciones, para interrogar por software la "línea del sensor (sense line)" que provee el cambio de estado de la existencia o no de la hoja de papel. Además no sería muy conveniente introducir en el programa principal una Subrutina de interrogación en forma muy frecuente, puesto que llevaría a que ese programa se "inflaría" demasiado y en la mayoría de los casos, la interrogación sería negativa, dado que la señal de interrupción no estaría activada, y la CPU trabajaría recargada inútilmente con la interrogación, no ejecutando las tareas del programa usuario. La técnica del polling es una solución económica y no utiliza recursos de hardware complejos, por lo que se lo instrumenta a través de varios métodos de utilización.

El procesamiento de la interrupción

Casi todos los Chips de microprocesadores tienen un pin de conexiones dedicados a la interrupción (interrupt), al cual se envía la señal IRQ de pedido que se tratará de acuerdo a la técnica que se use. Internamente se dispone de un conjunto de circuitos especializados para tratar las interrupciones que comúnmente se lo designa como **First Level Interrupt Handler** (FLIH), o sea un manejador de interrupciones de primer nivel.

En el tratamiento de la interrupción en un sistema rígido (hardware), el microprocesador verifica el ***estado*** del pin (INT = INTERRUPT), después de cada ciclo de instrucción. Si la señal INT está

inactiva, continúa con la próxima instrucción del programa que se está ejecutando en forma normal, cuyo proceso es un ciclo de máquina para la búsqueda (FETCH), otro ciclo para la lectura, otro para la descodificación, etc. hasta terminar con todas las operaciones codificadas en esa instrucción. En el siguiente esquema se observa como cada instrucción está dividido en ciclos de máquina y cada ciclo de máquina, a su vez, en ciclos de Reloj.

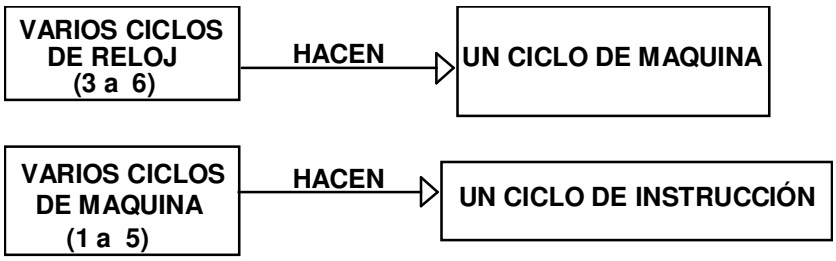


Fig. 1.A.4. Relación entre ciclos de reloj, maquina e instrucción

Si la señal INT está activa, entonces a en el próximo ciclo de instrucción comenzará con un ciclo de máquina de reconocimiento de la Interrupción (INTERRUPT ACKNOWLEDGE), como se observa en la figura 1.A.6.. Este ciclo es muy semejante al ciclo de búsqueda y ejecución normal, puesto que la CPU también tiene que ir a buscar una instrucción (compuesta por el Código de Operación), pero en vez de hacerlo en Memoria, lo hace desde un controlador de Periférico o de un controlador de Interrupciones. Para ello, el microprocesador, en vez de enviar al bus de control, la señal de LECTURA DE MEMORIA, coloca una señal de "Reconocimiento de Interrupción" (INTA-INTERRUPT-ACKNOWLEDGE), en ese bus.

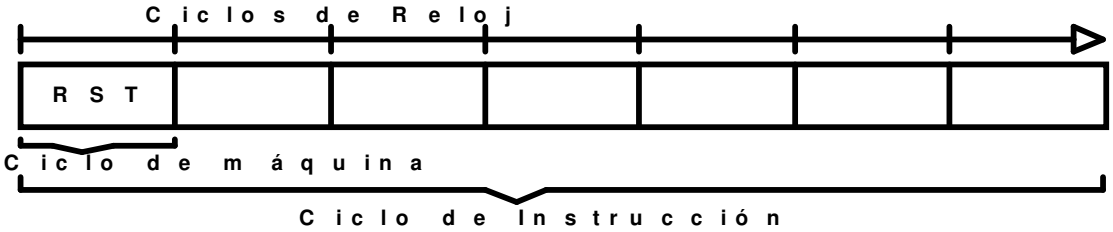


Fig. 1.A.5. Secuencia entre ciclos de reloj, maquina e instrucción

Esta señal INT se conecta a un dispositivo de control de interrupción, el que tiene una instrucción CALL con su correspondiente dirección de la subrutina (en el dibujo de la figura 1.A.8, representado por 3 bits que permite un vector de 8 tipos de tratamiento de Interrupciones), ó también puede ser una instrucción de Restart (RST), sobre el bus de datos. El bus de microprocesador lee éste código de operaciones que está colocado en el bus de datos, lo interpreta y lo ejecuta como si fuera una instrucción del programa principal en ejecución almacenado en memoria central.

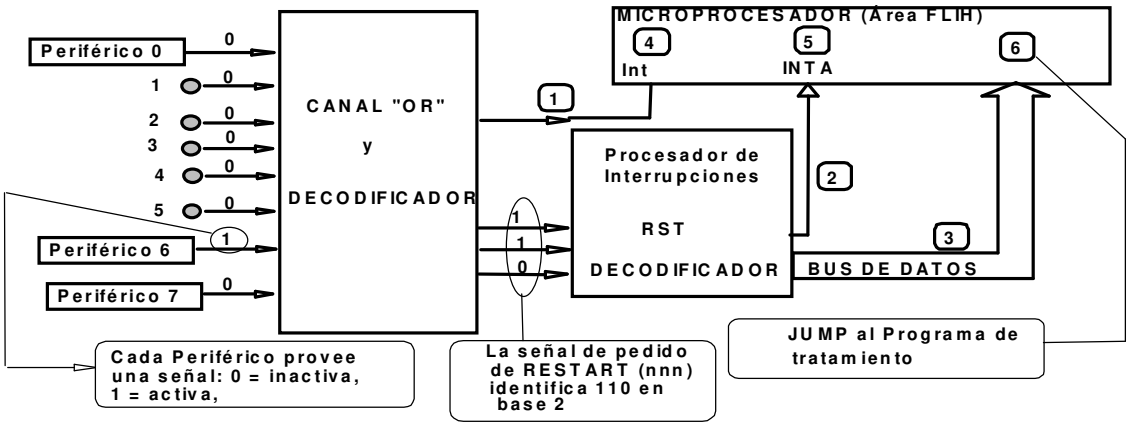


Fig. 1.A.6. Instancias en el procesamiento de la interrupción

CALL y RST actúan en forma semejante como fuera descripto en la lógica de respuesta sobre un pedido de interrupción. Cada salto, ya sea a una dirección dada o a una subrutina, primero de salvarse el contenido de los registros, mediante una instrucción PUSH, en un área de la Memoria Central reservada para el Sistema Operativo llamada ***área de pila o stack***, donde además queda guardada la dirección de la próxima instrucción del programa que deba ejecutarse a continuación, después de que se haya ejecutado la Subrutina de interrupción causante de la detención del programa principal. El estado del Contador de Programa se conoce como "Dirección de Retorno" (Fig.1.a.7), y es la que queda guardada en el Área de Pila

Supongamos que utilicemos un microprocesador de 8 bit (por ejemplo el 8085 de INTEL), en el primer ciclo de máquina (correspondiente a INTA), reconoce una instrucción CALL, entonces genera dos ciclos de máquina más, también correspondientes a INTA. Con estos ciclos se busca una dirección de salto para CALL. Esto genera un código, llamado CALL-CODE, con 2 Byte de longitud para una dirección de salto a un controlador de interrupciones que provee la información necesaria para resolver la interrupción. Se han diseñado y construido diversos Circuitos Integrados (Chip), que generan estos CALL-CODE (por ejemplo el 8259 de INTEL).

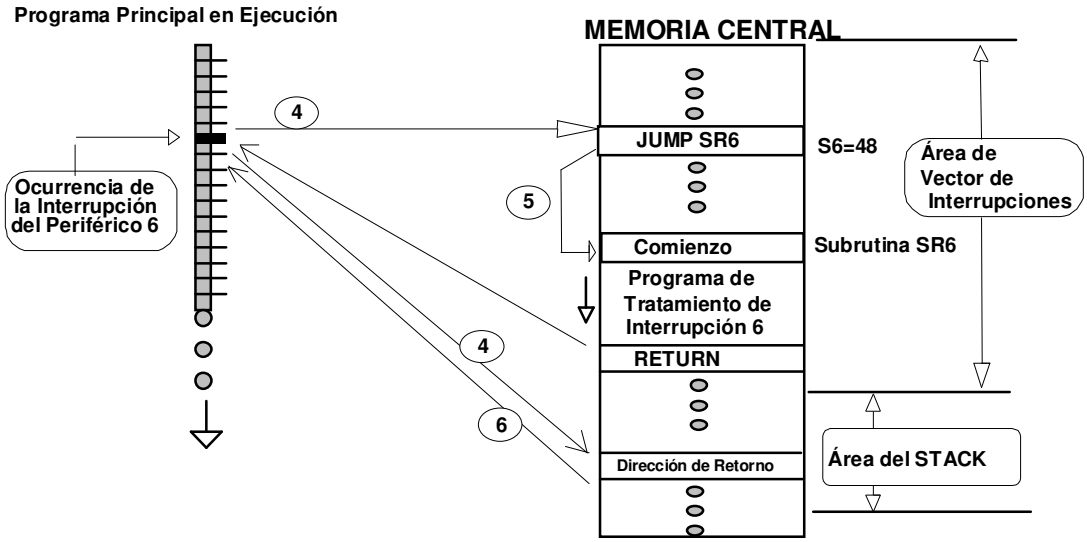


Fig. 1.A.7. Secuencia en el tratamiento de una interrupción

La generación en código de RESTART (RST-CODE), es muy sencilla. El Código de máquina de ésta instrucción tiene la configuración exhibida en la siguiente figura para el microprocesador 8085:

1	1	n	n	n	1	1	1

Fig. 1.A.8. Código de operación de una interrupción

Se trata de una instrucción cuya longitud es de un Byte donde "nnn" es un número más de RESTART de 3 bits. Con la combinación de estos 3 bits se puede asignar hasta 8 fuentes (periféricos), que generan señales de interrupción y que debe ser tratados cada uno en forma distinta.

En el tratamiento de la Instrucción RST, el microprocesador calcula la dirección de salto de la forma: valor de nnn por 8. Armándose así una pila de salto desde la dirección cero hasta la 64, con una separación constante de 8. Por supuesto que con 8 posiciones no se puede construir un programa completo que trate la interrupción, pero se puede solucionar introduciendo en esos lugares una instrucción de salto al comienzo de una subrutina de tratamiento de la interrupción, como se puede observar en la figura anterior.

Vector de Interrupciones:

Las instrucciones CALL ó RST en el ciclo de máquina durante INTA producen un salto direccionado a un vector de interrupciones que apunta a una dirección fijada por la fuente que produjo la interrupción. Esta dirección determina claramente el tipo de interrupción y su tratamiento o atención (VECTORED-INTERRUPT). Con un código RST se puede señalizar a 8 vectores distintos de interrupción y dentro de cada vector 8 niveles de interrupción.

En el caso del Circuito Integrado 8259 (Interrupt-Controller), puede controlar y administrar hasta 64 vectores distintos de interrupciones mediante un sistema en cascada. Además tiene un controlador de Prioridad de interrupciones y puede enmascarar a las interrupciones, con lo que cierra o libera las entradas de pedidos de interrupciones. Algunos microprocesadores también tienen incorporados en la Unidad de Control un controlador de interrupciones propio con capacidad para administrar distintos niveles.

Sistema mixto de interrupt-polling.

Consiste en la implementación por hardware del pedido de interrupción y el tratamiento realizarlo por software. Supongamos que se conecten 8 líneas de pedido de interrupción a una compuerta OR y la salida de la compuerta al microprocesador (pin INT). Al ocurrir una señal de pedido de interrupción en cualquiera de las 8 líneas, inmediatamente se entera el Procesador que genera una señal INTA que coloca siempre el mismo vector sobre el bus de datos, entonces el microprocesador va a inicializar un programa de tratamiento de interrupciones independiente del periférico que generó el pedido. Este programa puede reconocer a través del polling, las distintas líneas de pedidos sobre un canal de entrada. Este reconocimiento implica detectar cual es el periférico que ha producido el pedido (pedido activo), y luego deberá producir un tratamiento adecuado a través de una rutina especifica.

Procesamiento de Interrupciones

Veamos el rol del procesador en el procesamiento de las interrupciones con más detalle. La ocurrencia de una interrupción gatilla un número de eventos, tanto en el procesador en hardware como en el software. La Figura 6.19 muestra una secuencia típica. Cuando un dispositivo de E/S completa una operación de E/S, la siguiente secuencia de eventos en hardware ocurre:

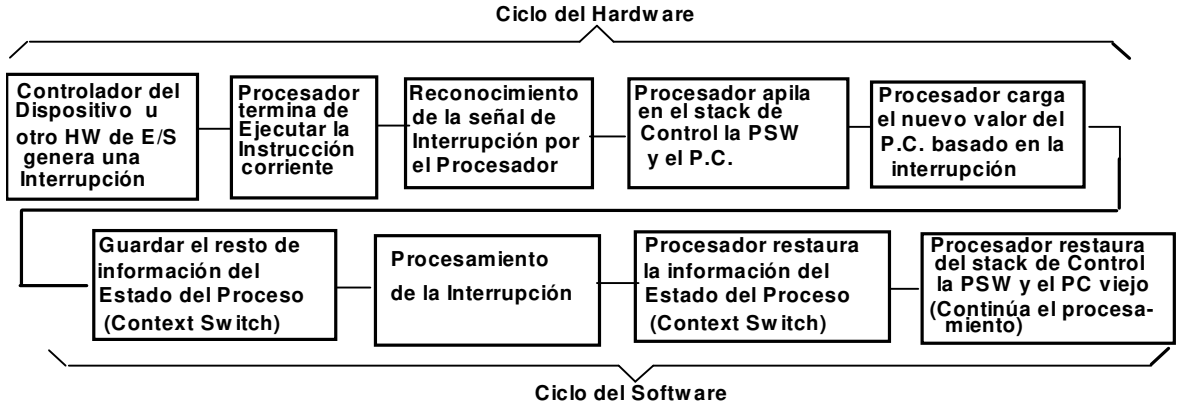


Fig. 1.A.9. Procesamiento de una interrupción

1. El dispositivo emite una señal de interrupción al procesador.
2. El procesador termina la ejecución de la instrucción corriente antes de responder a la interrupción, como se indica en la Figura 1.A.9.
3. El procesador verifica la ocurrencia de una interrupción, determina que hay una y envía una señal de reconocimiento (acknowledge), al dispositivo que emitió la interrupción. El reconocimiento permite al dispositivo remover su señal de interrupción (Figura 1.A.9.).
4. El procesador necesita prepararse ahora para transferir el control a la rutina que atiende la interrupción. Para empezar, necesita salvar la información necesaria para poder continuar el

programa en el punto en que fue interrumpido. La información mínima requerida es (a) el status del procesador, el cual está contenido en un registro llamado *program status word* (PSW), y (b) la dirección de la próxima instrucción a ser ejecutada, que está contenida en el registro contador de programa. Estos registros pueden ser apilados en el stack de control del sistema.

5. El procesador carga ahora el contador de programa con la dirección de entrada a la rutina que maneja la interrupción. Dependiendo de la arquitectura de computador y del diseño del sistema operativo, puede haber un único programa para cada tipo de interrupción, o uno para cada tipo de dispositivo y cada tipo de interrupción. Si hay más de una rutina que atienda la interrupción, el procesador debe determinar cual de ellas debe invocar. Esta información puede haber sido incluida en la señal de interrupción original, o el procesador debe emitir una solicitud al dispositivo que emitió la interrupción para obtener una respuesta que contenga la información necesaria.
   Una vez que el contador de programa haya sido cargado, el procesador procede con el ciclo de próxima instrucción, el cual comienza con obtener la instrucción (instruction-fetch). Dado que la obtención de la instrucción es determinada por el contenido del contador de programa, el resultado es que el control es transferido al programa que atiende la interrupción. La ejecución de este programa resulta en las siguientes operaciones:

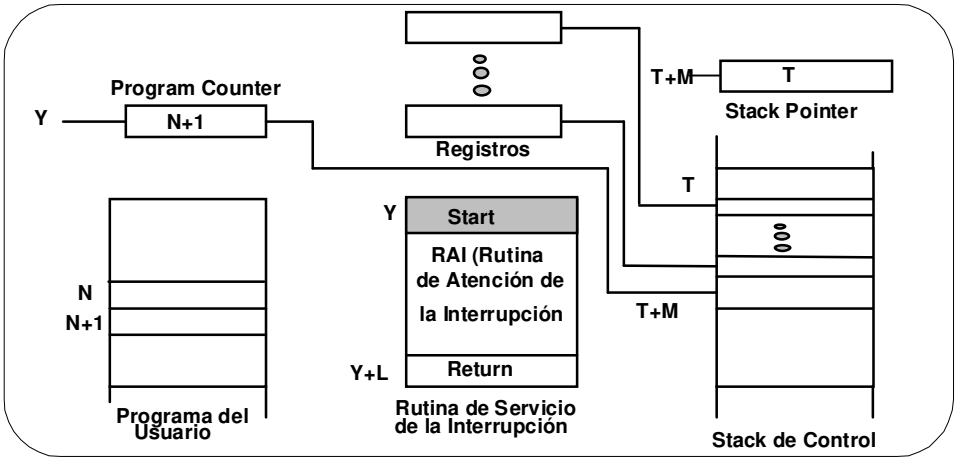


Figura 1.A.10-a. Procesamiento de una interrupción en la posición N

6. A este punto, el contador de programa y la PSW relativa al programa interrumpido han sido salvados en el stack del sistema. Sin embargo, hay otra información que es considerada como parte del "estado" del programa ejecutante. En particular, el contenido de los registros del procesador deben ser salvados, ya que podrían ser utilizados por la rutina de interrupción (First Level Interrupt Handler o simplemente handler). Así, todos estos valores, más toda otra información de estado, deben ser salvados. Típicamente, el handler de la interrupción comenzará salvando el contenido de todos los registros en el stack. La Figura 1.A.10. a muestra un ejemplo sencillo. En este caso, el programa usuario es interrumpido después de la instrucción en la posición N. El contenido de todos los registros más la dirección de próxima instrucción (N+1), son apilados en el stack. El puntero del stack se actualiza para apuntar el tope del stack, y el contador de programa se actualiza para apuntar al comienzo de la rutina de servicio de la interrupción. Actualiza significa que se cargan con los respectivos valores.
7. El handler de la interrupción puede ahora proceder a procesar la interrupción. Esto incluirá la examinación de la información de status relativa a la operación de E/S u otro evento que haya causado la interrupción. Esto puede también incluir el enviar comandos adicionales o reconocimientos (acknowledge), al dispositivo de E/S.
8. Cuando el procesamiento de la interrupción se completa, los registros salvados son desapilados del stack y recuperados en los registros (por ejemplo, ver Figura 1.A.10.-b).

9. El acto final es recuperar la PSW y el contador de programa del stack. Como resultado, la próxima instrucción a ser ejecutada será del programa previamente interrumpido u otro que determine el S.O..

Observe que es importante salvar toda la información de estado acerca del programa interrumpido para su posterior continuación. Esto es así porque la interrupción no es una rutina invocada desde el programa. En vez, la interrupción puede ocurrir en cualquier instante y en consecuencia en cualquier punto de la ejecución de un programa usuario. Su ocurrencia no es predecible. Más aún, los dos programas pueden no tener nada en común y pueden pertenecer a diferentes usuarios.

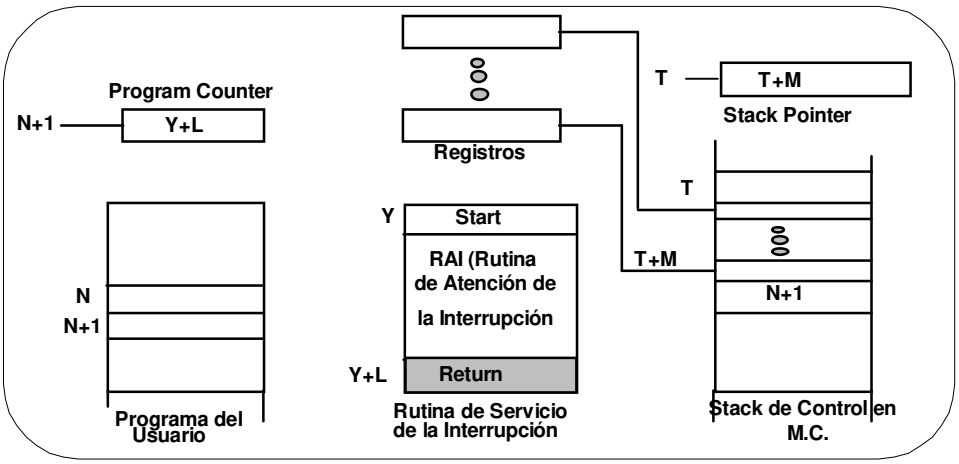


Fig. 1.A.10. -b Procesamiento de una interrupción (Retorno al Proceso interrumpido)

Aspectos de Diseño de interrupciones de periféricos:

Dos aspectos del diseño aparecen al implementar E/S por interrupciones. Primero, dado que habrán invariablemente múltiples módulos de E/S, ¿cómo determina el procesador qué dispositivo emitió la interrupción?. y segundo, si múltiples interrupciones ocurren, ¿cómo el procesador decide cuál procesar?.

Consideremos primero la identificación del dispositivo. Hay cuatro categorías generales de técnicas de uso común:

- Múltiples líneas de interrupciones.
- Interrogación (Polling): por software.
- Conexión circular (Daisy Chain): interrogación (poll), por hardware (vectorizado).
- Arbitración del bus (vectorizado).

La forma más directa para solucionar el problema es proveer múltiples líneas de interrupción entre el procesador y los módulos de E/S. Sin embargo, resulta impráctico dedicar más de unas pocas líneas de bus o pines del procesador a líneas de interrupción. Consecuentemente, aún si son utilizadas múltiples líneas, seguramente cada línea tendrá múltiples módulos de E/S conectados.

Una alternativa es la ***interrogación por software.*** Cuando el procesador detecta una interrupción, salta a la rutina de servicio de la interrupción cuyo trabajo es interrogar a cada módulo de E/S para determinar cuál de ellos causó la interrupción. La interrogación podría ser realizada en la forma de una línea de comando separada (por ejemplo TEST_I/O()). En este caso, el procesador produce el comando TEST_I/O y coloca la dirección de un módulo de E/S particular en las líneas de dirección. El módulo de E/S responde positivamente si éste había causado la interrupción.

Alternativamente, cada módulo podría tener un registro de status direccionable. El procesador luego lee el registro de cada módulo de E/S para identificar el módulo que interrumpió. Una vez que el módulo correcto es identificado, el procesador salta a la rutina de servicio específica del dispositivo.

La desventaja de la interrogación por software es el consumo de tiempo. Una técnica más eficiente es utilizar daisy-chain, la cual provee una interrogación por hardware. Para las interrupciones,

todos los módulos de E/S tienen una línea común de solicitud de interrupción. La línea de reconocimiento de interrupción es conectada circularmente (daisy-chained), a través de los módulos. Cuando el procesador censa una interrupción, transmite un reconocimiento de la interrupción. Esta señal se propaga a través de una serie de módulos de E/S hasta que alcanza al módulo que causó la interrupción. El módulo solicitante de la interrupción típicamente responde colocando una palabra en el bus de datos. Esta palabra se la refiere como un *vector* y es o bien la dirección del módulo de E/S o algún otro tipo de identificador único. En cualquier caso, el procesador utiliza el vector como un puntero a la rutina de servicio de atención de la interrupción apropiada. Esto evita la necesidad de ejecutar primero una rutina general de servicio de interrupción. Esta técnica se llama *interrupción vectorizada*.

Hay otra técnica que utiliza interrupciones vectorizadas, y es la ***arbitración de bus***. En ésta técnica, el módulo de E/S primero debe obtener el control del bus antes de que pueda "levantar" la línea que causó la interrupción. En consecuencia sólo un módulo a la vez puede levantar la línea. Cuando el procesador detecta la interrupción, responde en la línea de reconocimiento de interrupción (acknowledge). El módulo solicitante entonces coloca su vector en el bus de datos.

Las técnicas listadas arriba sirven para identificar el módulo de E/S solicitante. También proveen una forma de asignar prioridades cuando más de un dispositivo están solicitando un servicio de interrupción. Con líneas múltiples, el procesador elige primero la línea con más alta prioridad. Con la interrogación por software, el orden en que son interrogados los módulos establece la prioridad. Similarmente, el orden en que están conectados los dispositivos en una conexión circular (daisy-chain), también determina su prioridad. Finalmente, la arbitración de bus puede emplear una esquema de prioridades.

Veremos ahora dos ejemplos de estructuras de interrupciones.

Procesador de Interrupciones 8259 A

El Intel 8086 dispone de una única línea para solicitar interrupciones (INTR - Interrupt Request), y una única línea para enviar el reconocimiento (INTA - Interrupt Acknowledge). Para permitir que el 8086 maneje flexiblemente una variedad de dispositivos y estructuras de prioridad, éste se configura normalmente con un arbitrador de interrupciones externo, el 8259A. Este chip es un autentico procesador de pedido de interrupciones. Los dispositivos externos son conectados al 8259A, el cual a su vez se conecta con el 8086.

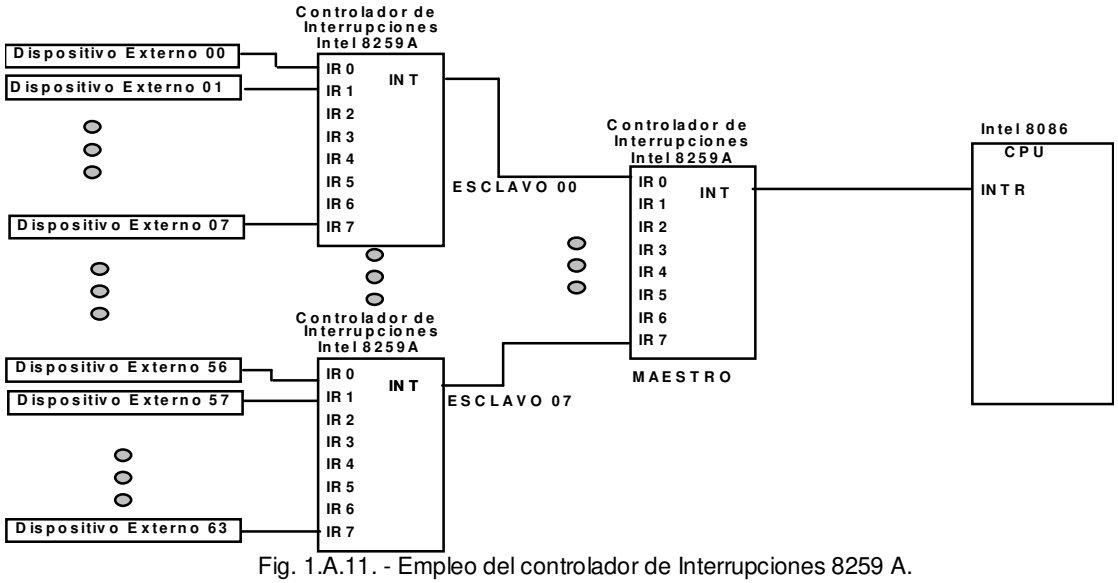


Fig. 1.A.11. - Empleo del controlador de Interrupciones 8259 A.

La Figura 1.A.11. muestra el uso del 8259A para conectar múltiples módulos de E/S para el 8086. Un único 8259A puede manejar hasta 8 módulos. Si se requiere el control de más de 8 módulos, un arreglo en cascada puede ser utilizado para manejar hasta 64 módulos.

La única responsabilidad del 8259A es la administración de las interrupciones. Acepta las interrupciones de los módulos a el conectado, determina cuál interrupción tiene la más alta prioridad, y luego señaliza al procesador levantando la línea INTR. El procesador responde vía la línea INTA. Esto provoca que el 8259A coloque la información de vector apropiada en el bus de datos. El procesador puede luego procesar la interrupción y comunicarse directamente con el módulo de E/S para leer o escribir datos

El 8259A es programable. EL 8086 determina el esquema de prioridades a ser utilizado al colocar la palabra de control en el 8259A. Los siguientes modos de interrupción son posibles.

- ***Anidado Completamente***: las solicitudes de interrupción están ordenadas en prioridad desde el 0 (IR0), hasta el 7 (IR7).
- ***Rotacional***: En algunas aplicaciones, un número determinado de dispositivos que interrumpen tienen la misma prioridad. En este modo, un dispositivo luego de haber sido servido, recibe la más baja prioridad en su grupo.
- ***Enmascarado especial***: Esto permite al procesador a inhibir selectivamente las interrupciones de ciertos dispositivos (y posteriormente habilitarlas)

La Interfase de Periféricos Programable Intel 8255

Como un ejemplo de un módulo de E/S utilizado para E/S programada y E/S por interrupción, vemos la interfase de periféricos programable 8255A.

El 8255A es un sólo chip, un módulo de E/S de propósito general diseñado para ser usado junto al procesador 8086 de Intel. La Figura 1.A.12. muestra un diagrama de bloques general donde se puede apreciar la interconexión del bus de datos y del bus de control.

El lado derecho del diagrama de bloques corresponde a la interfase externa del 8255A. Las 24 líneas de E/S son programables por el 8086 por medio de un registro de control. El 8086 puede establecer los valores del registro de control para especificar una variedad de modos de operación y configuraciones. Las 24 líneas son divididas en tres grupos de 8 bits (A, B y C). Cada grupo puede funcionar como un puerto de 8 bits de E/S. Además el Grupo C es subdividido en dos grupos de 4 bits cada uno (C_A y C_B), los cuales pueden ser utilizados conjuntamente con los puertos A y B. Configurados de ésta manera, estos subgrupos llevan señales de control y de status.

El lado izquierdo del diagrama de bloques es la interfase interna al bus del 8086. Incluye un bus de datos bidireccional de 8 bits (D0 hasta D7), el cual es utilizado para transferir los datos desde y hacia los puertos de E/S y para transferir información de control al registro de control. Una transferencia toma lugar cuando la línea CHIP SELECT está habilitada junto a la línea READ o a la línea WRITE. La línea RESET se usa para inicializar el módulo.

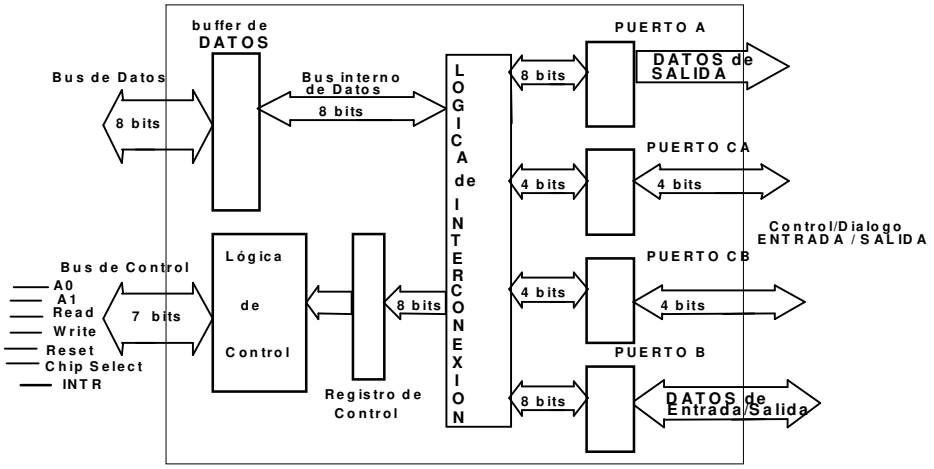


Fig. 1.A.12.- La Interfase de Periféricos Programable Intel 8255A

El registro de control es cargado por el procesador para controlar el modo de operación y para definir señales, si hay alguna. En la operación Modo 0, los tres grupos de 8 líneas operan como 3 puertos de 8 bits. Cada puerto puede establecerse como de entrada o de salida. De otra forma, los grupos A y B funcionan como puertos de E/S, y las líneas del grupo C sirven como líneas de control para los grupos A y B.

Las señales de control sirven a dos propósitos principales: "handshaking" y solicitud de interrupción. Handshaking (apretón de manos, darse la mano), es un mecanismo simple de tiempo. Una línea de control se utiliza por el transmisor como una línea DATA READY, para indicar cuándo un dato está presente en las líneas de E/S. Otra línea se usa por el receptor como un ACKNOWLEDGE, indicando que el dato ha sido leído y que las líneas de datos pueden ser limpiadas. Otra línea puede ser designada como INTERRUPT REQUEST y conectadas al bus del sistema.

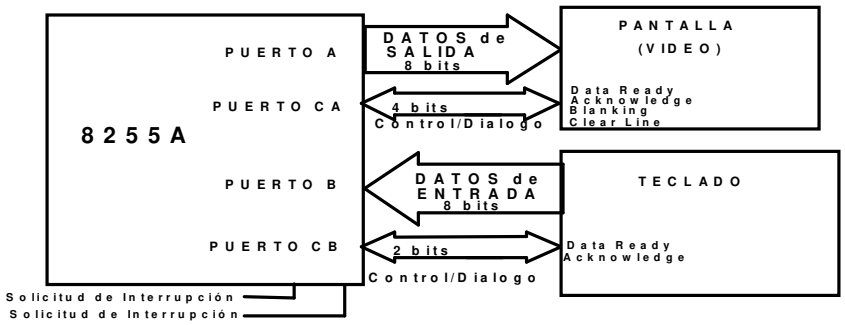


Fig. 1.A.13.- Interfase de Teclado / Pantalla al 8255A

Dado que el 8255A es programable vía el registro de control, puede ser utilizado para controlar una variedad de dispositivos periféricos simples. La Figura 1.A.13. ilustra su uso para controlar una terminal (pantalla y teclado). El teclado provee 8 bits de input. Dos de estos bits, SHIFT y CONTROL, tienen significado especial para el programa que maneja el teclado ejecutándose en la CPU. Sin embargo, ésta interpretación es transparente para el 8255A, el cual, simplemente acepta 8 bits de datos y los presenta en el bus de datos del sistema. Dos líneas de control del handshaking son provistas para usar con el teclado.

La pantalla está también enlazada a un puerto de 8 bits. Otra vez, 2 de éstos bits tienen significado especial y es transparente al 8255A. Además de las dos líneas para controlar el handshaking, se agregan otras dos para funciones de control adicionales.

Conclusión sobre el tratamiento por hardware de las interrupciones

De acuerdo a lo explicado, el sistema mixto provee un buen desempeño si se interroga con un intervalo de 500 o 1000 instrucciones dependiendo de la respuesta que provee el procesamiento de las interrupciones o sus prioridades.

Definimos como **tiempo de respuesta** al tiempo que transcurre desde que una señal de pedido se ha producido hasta que se ha comenzado el tratamiento de esa interrupción por su programa.

Suponiendo que el tratamiento se realiza cada 1000 instrucciones y que en promedio cada instrucción consume 10 pulsos de reloj, y cada pulso de reloj duran 5 nanosegundos (Clock de 200 MHz). En el peor de los casos tardaría 5 microsegundos en comenzar el tratamiento que sería el tiempo de respuesta. Si se reduce la cantidad de instrucciones se mejora el tiempo pero se produce un mayor procesamiento del polling reduciendo la velocidad de ejecución del programa del usuario.

Reconocimiento del PAI puede ser de dos estilos o formas:

- Polling (encuesta o escrutinio): Consulta periódicamente si se produjo la señal del PAI. Si la consulta resulta afirmativa entonces verifica a todos los dispositivos de E/S para detectar cual solicitó la señal y ejecuta la *Rutina de Atención de la Interrupción (RAI)*, para brindarle el servicio adecuado. Esta rutina se efectúa invocando al S.O.
- Interrupt Vector System (Vector de Interrupciones): El procesador recibe dos informaciones del Procesador de interrupciones: por una vía cableada le llega una señal de ocurrencia de

una interrupción. Decíamos que esta señal la denominamos PAI o en inglés *Interrupt request (IRQ).*

b) Tratamiento de la Interrupción por Software

Cuando se habla de multiprogramación, es porque hay varios procesos que se encuentran en estado *"activo" o de "ejecución"* ya que han sido comenzados pero aun no han sido completados o terminados. La máquina almacena estos programas en la memoria central. En una Computadora personal no existe la multiprogramación debido a que generalmente el S.O. es monousuario.

Recordemos que habíamos definimos como interrupción al "mecanismo" por el cual el procesador es forzado a reconocer la ocurrencia de un evento en el sistema mediante una señal.

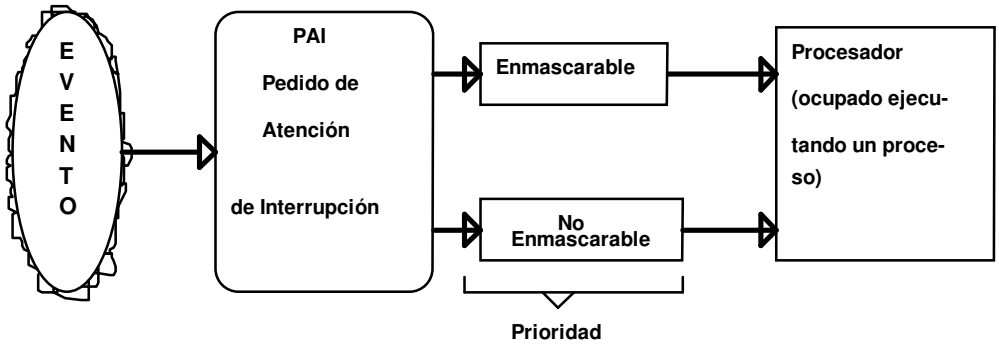


Fig. 1.A.14. Ocurrencia de un PAI

También es aplicable a la detención de un proceso que está usando CPU debido a condiciones externas al procesador.

Ejemplo de causales de interrupciones: Mal funcionamiento del hardware (HW), Finalización de E/S (HW), Time-out de un reloj (HW), interrupción de teclado (HW), Comienzo de E/S (SW), System Calls (SW), etc.,

Todo acceso a memoria suele tener un hardware de protección. Cuando se intercepta un acceso que no corresponde o es indebido, se produce una interrupción y se lo trata como un error.

El Hardware debe proveer un "Mecanismo de interrupción": Esta condición es fundamental para el procesamiento por eventos y normalmente hay destinado todo un complejo hardware para esta función. Hay procesadores en chips que se ocupan de las interfases de las señales que son originados por los eventos y procesan los Pedidos de *Atención de una Interrupción (PAI)*, o *Interrupt request (IRQ).* Este pedido debe ser conectado al procesador central quien debe reconocer a ésta señal *(interrupt aknowledge* o INTA), mediante un mecanismo denominado *Gestor de interrupciones de primer nivel (First Leve/ Interrupt Handler - FLIH)*. De acuerdo al modo de funcionamiento del procesador se enmascarará o no al PAI.

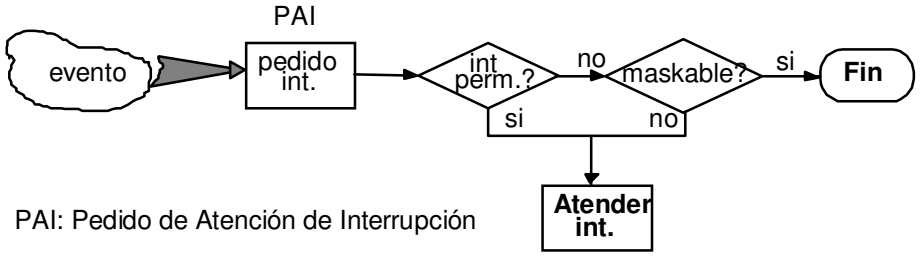


Fig. 1.A.15. Proceso de prioridades de una interrupción

Recibe mediante una lectura del Bus de Datos un Código de Operación en el que el Procesador de Interrupciones ha incorporado, mediante un direccionamiento inmediato, el código del dispositivo que produjo el PAI. Con éste código la CPU direcciona una posición de Memoria Central en que se encuentra la entrada a una tabla *(Vector de Interrupción)*, que contiene la dirección de Memoria en que

se aloja la Rutina de Atención de esa interrupción (RAI), que va a ser la responsable de brindar el servicio solicitado por el PAI.

Cada entrada de la tabla indica el tipo de dispositivo, su dirección y su estado (si el estado es ocupado, el tipo de solicitud y otros parámetros se anotan en tabla del dispositivo). Pueden efectuarse varias solicitudes PAI para el mismo dispositivo, y para esto el S.O. cuenta con una "lista de solicitudes" en espera.

Las interrupciones pueden ser desactivadas mientras una de ellas se procesa, demorando la interrupción que entra hasta que el S.O. termine con la actual. Esto se conoce como enmascaramiento de interrupciones. El S.O. lleva el control de éstos Pedidos de Atención y los va procesando (activando), a cada una de las que están esperando hasta que todas las interrupciones se hayan activado.

Si no se desactivaran, la última escribiría sobre los parámetros y datos de la primera y sería una *interrupción perdida.*

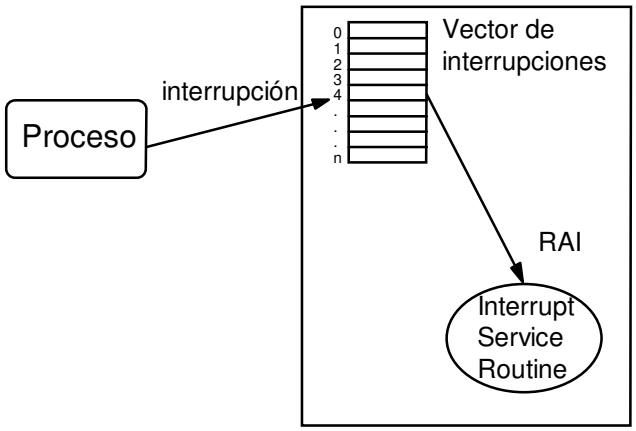
Las arquitecturas complejas permiten ejecutarlas simultáneamente siguiendo esquemas donde se les asigna prioridades a los tipos de solicitud de acuerdo con importancia relativa y se almacena por separado la información de procesamiento de la interrupción para cada prioridad. Las interrupciones de igual o menor prioridad se enmascaran o desactivan selectivamente, para evitar interrupciones perdidas o que no sirven.

La implementación de cada Llamada al Sistema (System Call), suele realizarse por medio de interrupciones.

Para la RAI se reservan un conjunto de posiciones en la parte baja de la memoria (generalmente las primeras 1000 posiciones), en las que se almacenan las direcciones de las rutinas (RAI), para los distintos dispositivos. Se carga la RAI reemplazando el contenido del Program Counter por las direcciones donde se encuentran la rutinas. Primero se determina el origen de la interrupción examinando el PAI, se carga la correspondiente rutina que atenderá a la interrupción y luego, con un salto a direcciones de retorno almacenadas, continuará la ejecución del proceso que estaba en uso del procesador u otro que decida el SO. Existen direcciones donde la CPU tiene los datos recibidos de algunos dispositivos. Decíamos que hay dos estrategias o procedimientos clásicos mediante los cuales la CPU atiende esas direcciones: el Polling y un Vector de interrupciones.

Para ampliar estos conceptos sobre el reconocimiento de la ocurrencia de una Interrupción ya explicamos los criterios para implementar estas estrategias:

- Polling, (Sincrónicamente - es una revisión cíclica de esas direcciones para saber que es lo que está pasando).
- Vector de Interrupción (Asincrónicamente).



RAI: Rutina de Atención de la Interrupción.

Fig. 1.A.16. Tratamiento de interrupciones por medio de un Vector

Dado que existe una cantidad limitada (y reducida), de tipos de interrupciones posibles, generalmente se usa una tabla con las direcciones de los servicios de cada interrupción, típicamente en las primeras posiciones de la memoria. Esta tabla se conoce como vector de interrupciones. Así por ejemplo, si las direcciones ocupan 4 bytes, la dirección del servicio de una interrupción se guarda en los 4 bytes a partir de la posición *múltiplo de 4* de la memoria.

Es importante que antes de atender a la interrupción se preserve los contenidos de los registros y cierta información acerca del estado del proceso que será interrumpido en el PCB o vector de estado (las arquitecturas más modernas almacenan el PCB en el STACK), esto se indica como guardar en Memoria mediante un *cambio de contexto (context swítch)*.

El sistema de Interrupciones (First Level Interrupt Handler), que dispone el Procesador, está asociado al Kernel mediante el juego de instrucciones de máquina específicas compuesto por:

- Llamadas al Sistema (System Call).
- Operaciones de E/S.
- Tiempo o interrupciones del reloj.
- Errores de Programación (program check).
- Page Fault (en paginación).
- Interrupciones externas.
- Otras.

Los tratamientos de Pedido de Atención de Interrupción o IRQ (Interrupt ReQuest), son importantes por los siguientes motivos:

- La rapidez con que se atiende a las interrupciones es crítica para ciertos dispositivos, porque de ellas depende su buen funcionamiento.
- El caso más importante es el del teclado. Cada vez que el usuario oprime una tecla se produce una interrupción que debe ser atendida antes de que se oprima otra.

IRQ	DISPOSITIVO
0	Reloj
1	Teclado
2	Pointer para los IRQ 8 a 15
3	COM2 / COM4
4	COM1 / COM1
5	Sin usar
6	Disco flexible
7	LPT1
8	Tiempo
9	Red LAN
10	Sin uso
11	Sin uso
12	Sin uso
13	Coprocesador
14	Disco duro
15	Sin uso

Tabla 1.A.1. Ejemplo de Interrupciones en una PC

Los PAI o INTERRUPT REQUEST son como señales, semejantes a un gong que suena y al que la CPU obedece, suspendiendo lo que está haciendo para abocarse a atender al pedido del recién llegado.

Las PC-XT solo tenían 8 niveles de interrupción. En cambio las PC-AT tienen hasta 16 niveles. En la Tabla 1.A.1 se detallan las interrupciones de un Computador Personal del tipo AT. Cada interrupción es ocupada por uno y sólo uno de los dispositivos porque si dos de ellos tuviesen una misma interrupción no podrían trabajar adecuadamente, por lo tanto si dos periféricos tuvieran una misma interrupción habría que definirle otra mediante switches, jumpers o llaves DIP en la propia tarjeta.

Aunque no todas las plaquetas tienen switches, jumpers o llaves DIP y eso es un inconveniente porque entonces habrá que suprimir una interrupción, la que se considere menos importante.

A continuación estudiaremos los diferentes tipos de interrupciones a saber :

- Externas (Consola del operador, teclado).
- Fin de E/S.
- Llamadas al SO.
- Fin (normal o anormal).
- E/S (en espera, o sin - sincronización -)
- Información.
- Errores.
- Reloj de intervalos.

Tener varios niveles de atención de interrupciones da la posibilidad de trabajar en modalidad multitarea.

La idea final de la parte de Sistemas Operativos es conocer el flujo de un programa a lo largo de toda su ejecución, sus estados, y las tablas y programas que intervienen en la vida del programa / trabajo / comando.

Interrupciones de Entrada y Salida: Son iniciadas por hardware de entrada y salida. Estas interrupciones indican a la UCP el cambio de estado de un canal o dispositivo. Las interrupciones de E/S se producen cuando finaliza una operación de E/S o cuando un dispositivo pasa al estado listo. **Interrupciones del procesador:** Una *interrupción* es un evento que altera la secuencia en que el procesador ejecuta las instrucciones. La interrupción es generada por el hardware del sistema de cómputo.

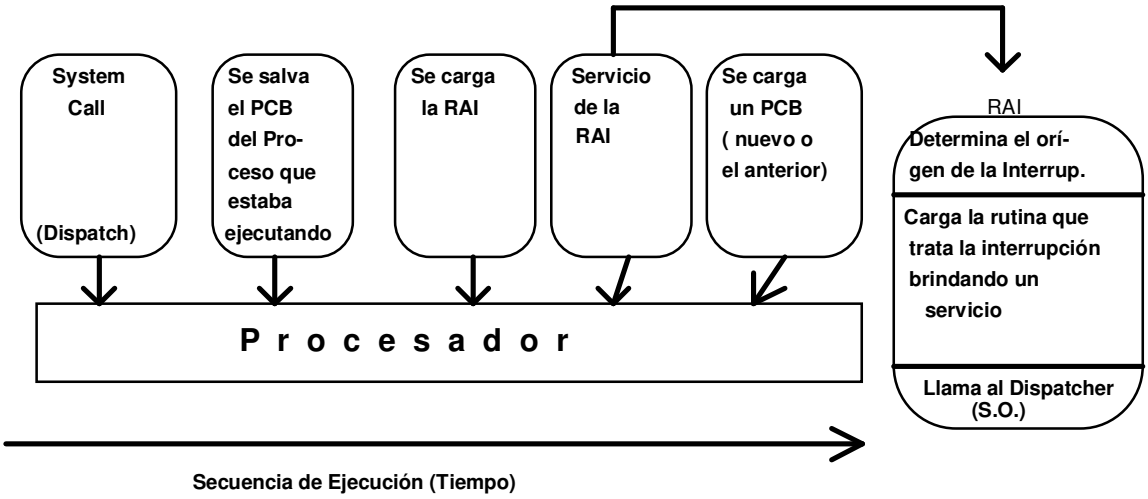


Fig. 1.A.17. Secuencia de ejecución de una RAI

Cuando ocurre una interrupción los pasos que se sigue son:

- El sistema operativo toma el control (es decir, el hardware pasa el control al sistema operativo).
- El sistema operativo guarda el estado del proceso interrumpido. En muchos sistemas esta información se guarda en el bloque de control de proceso interrumpido.
- El sistema operativo analiza la interrupción y transfiere el control a la rutina apropiada para atenderla; en muchos sistemas actuales el hardware se encarga de esto automáticamente.

- La rutina del *manejador de interrupciones (Interrupt Handler)*, procesa la interrupción.
- Se restablece el estado del proceso interrumpido (o del "siguiente proceso").
- Se ejecuta el proceso interrumpido (o el "siguiente proceso").

Esquemas más sofisticados *enmascaran* (deshabilitan selectivamente), las interrupciones en vez de deshabilitarlas totalmente, de manera que una interrupción de alta prioridad SI pueda interrumpir al procesador cuando está atendiendo una de menor prioridad.

Una interrupción puede ser iniciada específicamente por un proceso en ejecución (se dice que está *sincronizada* con la operación del proceso), o puede ser causada por algún evento que puede estar relacionado o no con el proceso en ejecución (en cuyo caso se dice que es *asíncrona* con la operación del proceso).Los sistemas orientados hacia las interrupciones pueden sobrecargarse.

Si estás llegan con mucha frecuencia, el sistema no será capaz de atenderlas. En algunos sistemas orientados hacia el teclado, cada tecla presionada almacena en la memoria un código de un byte y genera un interrupción para informar al procesador que un carácter está listo para ser procesado. Si el procesador no puede procesar el dato antes de que se presione la siguiente tecla, se pierde el primer carácter.

Clases de Interrupciones tratadas por el S.O.:

Existen seis clases de interrupciones a saber:

- **Interrupciones System Call** (supervisor call- SVC, llamadas al supervisor). Son iniciadas por un proceso en ejecución que ejecute la instrucción system call. Un system call es una petición generada por el programa en ejecución que requiere un servicio particular del sistema, por ejemplo, realizar una operación de entrada/salida, obtener más memoria o comunicarse con el operador del sistema. El mecanismo de los system call ayuda a proteger el sistema operativo de las acciones de los usuarios. Un usuario no puede entrar arbitrariamente al sistema operativo, sino que debe solicitar un servicio por medio de un system call. El sistema operativo está al tanto de todos los usuarios que intentan rebasar sus límites y puede rechazar ciertas peticiones si el usuario no tiene los privilegios necesarios.
- **Interrupciones de E/S.** Son iniciadas por hardware de entrada y salida. Estas interrupciones indican al procesador el cambio de estado de un canal o dispositivo. Las interrupciones de E/S se producen cuando finaliza una operación de E/S o cuando un dispositivo pasa al estado listo.
- **Interrupciones externas.** Son causadas por diversos eventos, incluyendo la expiración de un dado tiempo de un reloj que interrumpe, la pulsación de la tecla de interrupción de la consola o la recepción de una señal procedente de otro procesador en un sistema de múltiples procesadores.
- **Interrupciones de Reinicio.** Se produce cuando se presiona el botón de reinicio de la máquina o cuando llega de otro procesador una instrucción de reinicio en un sistema de multiprocesamiento.
- **Interrupciones de excepción del programa también llamadas trap.** Son causadas por una amplia clase de problemas que pueden ocurrir cuando se ejecutan las instrucciones en lenguaje máquina de un programa. Dichos problemas incluyen la división por cero, el exceso o defecto de los números que pueden ser manejados por las operaciones aritméticas, el intento de hacer referencia a una localidad de memoria que esté fuera de los límites de la memoria real. Muchos sistemas ofrecen a los usuarios la opción de especificar las rutinas que deben ejecutarse cuando ocurra una interrupción de excepción del programa.
- **Interrupciones de verificación de la máquina.** Son ocasionadas por el mal funcionamiento del hardware. Fallas o defectos.

Las Interrupciones son la base de los sistemas operativos modernos, sin ellos no se puede concebir una ejecución de programas.

Un computador moderno se compone de un procesador(a veces más de uno), y una serie de

controladores de dispositivos, conectados a través de un bus común a la memoria. Cada controlador está a cargo de un tipo específico de dispositivo.

Cuando se enciende el computador, se ejecuta automáticamente un pequeño programa, cuya única tarea es cargar el sistema operativo en la memoria, y entregarle el control (ejecutarlo). El sistema operativo hace las inicializaciones del caso, y luego simplemente espera a que algún evento ocurra.

La ocurrencia de un evento es, por lo general, señalizada por una interrupción de software o hardware. Los controladores de dispositivo pueden generar una interrupción en cualquier momento, enviando una señal al procesador a través del bus del sistema. Las interrupciones de software son generadas por los procesos, por la vía de ejecutar una instrucción especial que se conoce como llamada al sistema.

Ejemplos de eventos que generan interrupciones: terminación de una operación de I/O, llamadas al sistema, división por cero, alarmas del reloj. Para cada tipo de interrupción, debe existir una rutina adecuada que la atienda y que es invocada por el S.O.

Todos los sistemas operativos modernos se basan en interrupciones (*interrupt driven*). Si no hay procesos que ejecutar, ni dispositivos ni usuarios que atender, el sistema operativo no hace nada, o sea espera que algo pase. Cuando algo pasa, se señaliza una interrupción, y el sistema operativo entra en actividad (o sea, las rutinas de servicio de interrupción son parte del sistema operativo).

Ejemplo: multiprogramación (timesharing)

- El sistema operativo inicializa el *timer* o reloj en una tajada de tiempo, y lo pone en marcha.
- El sistema operativo entrega el control a un proceso.
- El proceso ejecuta.
- Concluido el tiempo prefijado, el timer provoca una interrupción.
- El manejador de interrupciones del timer (que es parte del sistema operativo), guarda la información del proceso interrumpido necesaria para poder reanudarlo después.
- Se repite el ciclo, seleccionando ahora otro proceso.para ejecutar

Otro ejemplo: manejo de dispositivos de I/O: Para comenzar una operación de I/O, el sistema operativo escribe los registros del caso en el controlador del dispositivo. A su vez, el controlador determina qué hacer mirando el contenido de esos registros. Por ejemplo, si se trata de una solicitud de lectura, inicia la transferencia de datos desde el dispositivo hacia la memoria. Cuando finaliza la transferencia, el controlador le avisa al procesador a través de una interrupción. Esto permite que el procesador, en vez de esperar que la transferencia termine (lo que sería E/S sincrónico), en el intervalo puede realizar otra tarea (E/S asincrónico). Una secuencia típica:

- El proceso que está ejecutando en el procesador solicita una operación de E/S, mediante una llamada al sistema.
- El sistema operativo suspende el proceso, poniéndolo en la cola de procesos suspendidos, y comienza la operación de I/O.
- El sistema operativo selecciona otro proceso para que siga ejecutando en el procesador, mientras la operación de E/S se completa.
- Cuando la operación de E/S se completa, el control vuelve al sistema operativo gracias a que el controlador del dispositivo provocó una interrupción.
- El sistema operativo, determina, según la interrupción y sus tablas internas, que el proceso que había sido suspendido ahora puede reanudarse, así que lo pone en la cola de procesos listos para ejecutar.
- El sistema operativo reanuda ese, o tal vez otro proceso de la cola de procesos listos.

RESUMEN sobre Interrupciones:

Una interrupción se produce ante las siguientes situaciones:

a) Interrupción por E/S: I/O completed - I/O error.
b) Programas: Invalid instruction, overflow, underflow, división por 0, system calls.
c) Otras: reloj, reset.

Generalmente se maneja con *puntos de seguridad (safe point)*, colocados dentro de los programas (se salva periódicamente el PCB en Memoria o en un soporte y ante la caída del sistema se

reanuda desde el último safe point grabado).

Hay diferentes eventos que pueden disparar una Interrupción y para cada uno hay un servicio (Rutina), de Atención.

Las interrupciones son una parte importante de la Arquitectura de una Computadora y cada diseño tiene su propio mecanismo de interrupciones, pero las funciones son comunes:

La interrupción debe transferir el control a la RAI apropiarla.

El método consiste en que la RAI examina el PAI e invoca la rutina específica.

La mayoría de los Computadores predefinen un Número de interrupciones posibles en una Tabla de punteros *(Vector de Interrupciones)*, almacenada en las locaciones bajas de la Memoria Central.

Las interrupciones pueden estructurarse en Niveles de Prioridad (las de menor prioridad se enmascaran ante las de mayor prioridad).

Anexo 1.B. Instrucciones

Diseño de un Computador básico:

El diseño de un computador digital es la organización de módulos de hardware relacionados por rutas de control y datos. Su función es permitir el flujo de señales binarias para transformar datos de entrada en información útil al usuario del computador.

Cada módulo realiza una o varias operaciones sobre datos codificados en binario, que se almacena en registros asociados al módulo, mientras dura la operación. Una operación aplicada a un registro se denomina *microoperación* y se activa en un instante de tiempo sincronizados por los tiempos del reloj.

Representación de datos a Nivel Máquina:

Tanto las instrucciones de los programas como los datos que ingresan a la memoria desde el exterior lo hacen en un código alfanumérico de representación de caracteres. Las instrucciones son transformadas por un programa de traducción o código de máquina que es el que entiende el procesador para luego ser ejecutadas.

Estas formas de almacenamiento pueden ser
- como código ASCII
- como BCD
- como binarios enteros(Binarios en punto fijo)
- como binarios reales (Binarios de punto flotante)

El almacenamiento en la memoria ya sea de datos o códigos de programas, sigue un orden específico. En la memoria el byte menos significativo se almacena en la dirección más baja, mientras el más significativo lo hace en la dirección más alta. Sin embargo hay que tener en cuenta, si el dato es numérico, cada octeto se almacena en memoria en forma invertida. Esto ocurre para cualquier entidad numérica, incluso para datos en representación en punto flotante o para direcciones en memoria.

Cuando la CPU direcciona un byte cuya dirección es par, se referencia al byte (n), y al impar que le sigue (n+1). Es decir que por cada orden de lectura se obtienen 16 bits en el databus. Si por el contrario se direcciona un bit cuya dirección es impar, se referencia al byte impar (n+1), y al par que la sigue (n+2).

Instrucciones:

Código de operación de una instrucción: Es la combinación de bits que la Unidad de Control interpreta para generar las microoperaciones que permitan su ejecución.

Formato de una instrucción:

La forma de agrupar estos bits determina la estructura de la instrucción. Cuando el usuario confecciona un programa utiliza un lenguaje simbólico más orientado a su modalidad de expresión que a la del computador y éste debe ser traducido a código de máquina antes de su ejecución, ya que solo ejecuta códigos de instrucción en lenguaje de máquina.

La ejecución de una instrucción implica una secuencia de movimientos de transferencia de bytes entre la MC y los registros de la CPU (o entre los últimos), establecidos por la UC en un orden determinado, según el código de dicha instrucción.

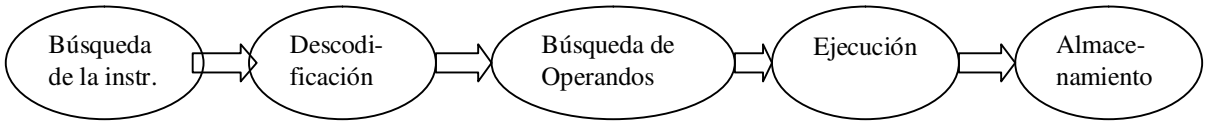


Fig. 1.B.1. Ciclo de una instrucción

Cuando la UC ejecuta un programa debe alternar sus etapas *fetch* y *execute* desde la primera instrucciones hasta la última. La secuencia del ciclo $f(i_n)$ $e(i_n)$ se denomina *Ciclo de la instrucción.*

Para cada fase f(in), la UC debe enviar a la memoria la dirección de la palabra donde se encuentra la instrucción, una orden de lectura, y una orden de transferencia de la misma a la UC. La UC retiene la dirección de la instrucción en un registro denominado PI (Puntero de instrucciones), ó CP (Contador de Programa). I_0 carga siempre en la palabra 000H, de modo que es este el valor inicial de la CPU cuando se arranca el funcionamiento del computador. Además las instrucciones se almacenan en palabras sucesivas, por esta razón el CP debe ser incrementado en una unidad para poder señalar siempre la próxima instrucció a ser buscada. El CP puede, también, aceptar un valor cualquiera impuesto por una instrucción que provoca una ruptura de secuencia en el programa.

Una vez que la UC envía a la memoria el contenido de CP, da la orden de lectura para que la palabra implicada, caiga en el registro de palabra de memoria. La etapa *fetch* termina con la transferencia de la instrucción leída a un registro interno de la UC, donde la instrucción permanece almacenada mientras dure su posterior tratamiento. Este registro se denomina *Registro de instrucciones.*

El registro de instrucciones está relacionado directamente con el hardware que genera microoperaciones desde la instrucción.

La zona denominada DATA se relaciona con el MAR cuando se debe tomar un dato de memoria y con el CP cuando se debe romper la secuencia normal del programa. Esta zona se ignora cuando la instrucción no afecta a dato alguno. La UC actualiza el CP y entra en la fase execute de la instrucción buscada. En un Hag llamado F (fase), se retiene un 1 durante la fase de búsqueda que cambia a 0 cuando comienza la fase de ejecución.

Referencias del gráfico adjuntado:

1- PC ⟶ MAR
2- W ⟶ MDR
3- MDR ⟶ IR
4- PC+1 ⟶ PC
5- 0 ⟶ F

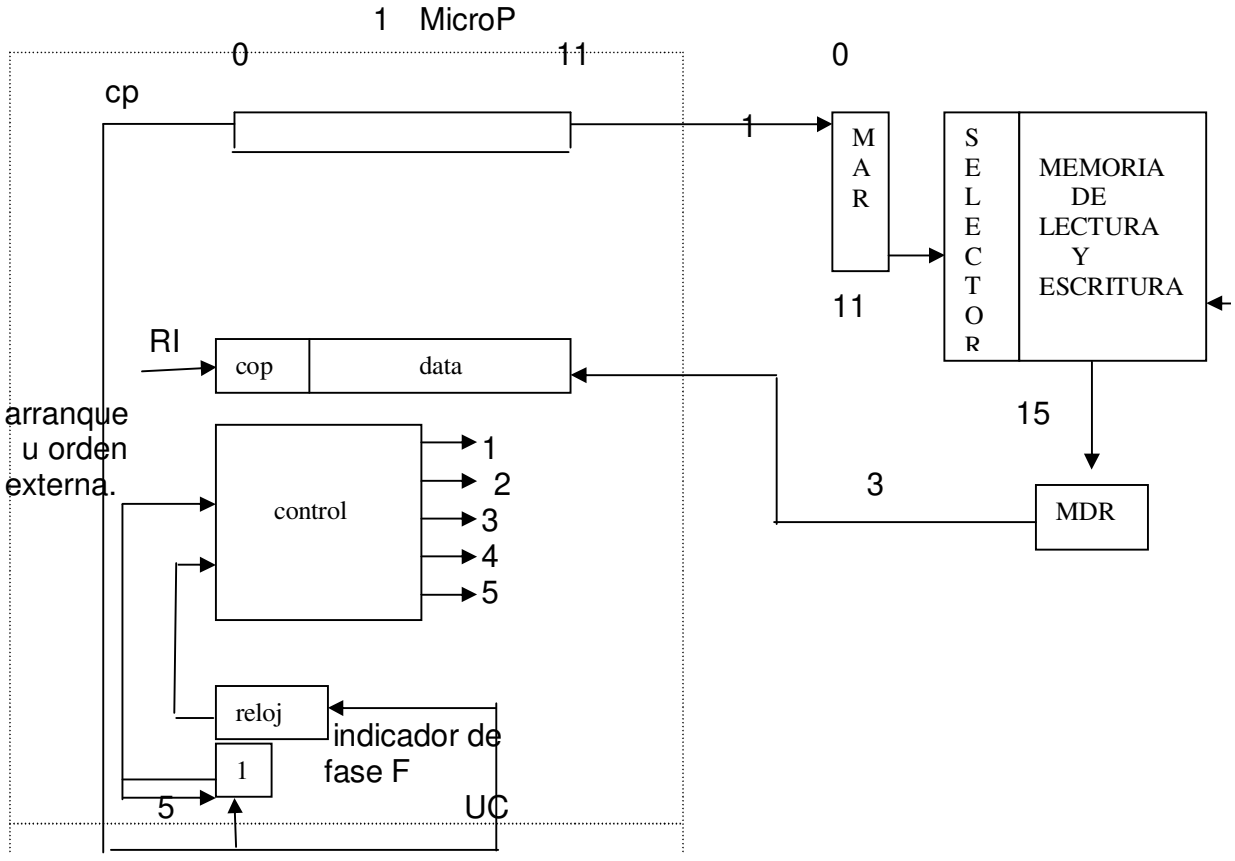


Fig. 1B.2

AUTOEVALUACIÓN DEL MODULO 1:

Preguntas:

1) La arquitectura por capas (layers) de Dijsktra determina que la capa (n-1) le abstrae ciertas funciones a la capa n, mientras que la capa (n+1) le entrega servicios a esta capa n. ¿Es verdad esto?. Justifique.
2) ¿ Es verdad que en cada ciclo de instrucción el sistema operativo se fija si aconteció alguna interrupción? Justifique.
3) Nombre las principales ventajas y desventajas de diseñar un sistema operativo usando una arquitectura de maquinas virtuales. ¿Cuales son las ventajas y desventajas para el usuario?
4) ¿ A qué áreas provee servicios un S.O?
5) ¿ Qué protección de la información y seguridad brinda un S.O.?
6) ¿Cuáles son las funciones de un Sistema Operativo?
7) Explicar las diferencias entre las interrupciones ENMASCARABLES y las NO ENMASCARABLES.
8) Nombrar los métodos básicos que utiliza el S.O. para brindar los servicios.

Múltiple Choice:

1.-La función de inicialización: a) Es parte del Kernel. b) Crea las estructura de datos del Sistema. c) Carga en memoria las RAI (rutinas de atención de Interrupción). d) Carga el BIOS. e) Cuando finaliza deja el shell activo. f) Todas las anteriores g) Ninguna de las anteriores	**2.- Los sistemas operativos con arquitectura cliente-servidor** a) Son más fáciles de comprender su diseño. b) Pueden transformar a un Servidor en Cliente y viceversa. c) Son más fáciles de mejorar y corregir sus problemas. d) Pueden seguir funcionando aunque parte del sistema se caiga. e) Implementan mejor la protección del hardware f) Todas las anteriores g) Ninguna de las anteriores
3. Los System call realizan: a) Servicios al usuario b) Servicios al S.O. c) Atención de interrupciones. d) Atención de excepciones. e) Creación y finalización de procesos. f) Todas las anteriores g) Ninguna de las anteriores	**4.- El kernel** a) Realiza la Planificación de CPU en el más bajo nivel. b) Cambia el modo de *ejecución* de/Procesador c) Amplía el set de instrucciones de la máquina d) Comunicación de Procesos e) Todas las anteriores f) Ninguna de las anteriores
5.- ¿Cuáles son las Funciones del Sistema Operativo? a) Comunicación de Procesos. b) Inicialización del Sistema. c) Administración de Procesos. d) Administración de Recursos. e) Protección. f) Brindar una interfase Gráfica a los usuarios. g) Todas las anteriores h) Ninguna de las anteriores	**6 - Un sistema operativo es:** a) Un Programa. b) Un compilador c) Un conjunto de rutinas d) Una biblioteca de funciones e) Todas las anteriores f) Ninguna de las anteriores
7.- ¿Cuáles son los objetivos fundamentales de los S.O.? a) Facilitar el trabajo al usuario. b) Ejecutar al procesador c) Gestionar en forma eficiente los recursos. d) Inicializar la máquina. e) Proteger a los usuarios y a los recursos. f) Todas las anteriores g) Ninguna de las anteriores	**8.- La función de inicialización del Sistema Operativo:** a) Carga el Kernel en la memoria central. b) Crea e inicializa las tablas para administrar recursos. c) Siempre ejecuta una rutina de verificación de recursos. d) Carga el Shell en la memoria central e) carga la BIOS en la ROM f) Todas las anteriores son correctas. g) Ninguna es correcta.
9.- En un sistema diseñado por capas: a) Cada capa puede pedir servicios a sus capas adyacentes. b) La organización de las capas responde a un Ordenamiento Jerárquico. c) Cada capa tiene información sobre la implementación de la capa inferior. d) Cada capa tiene información sobre la implementación de la capa superior. e) Es difícil de depurar su programación	**10.- El Shell:** a) Tiene como principal objetivo ocultar el complejo funcionamiento del sistema al usuario. b) En ciertas ocasiones, es capaz de planificar los pedidos de E/S. c) Se puede dividir en dos grupos bien marcados: los que tienen interfase basada en comandos y los de interfase gráfica. d) Se ocupa de inicializar la máquina e) Son funciones programadas

f) Todas las anteriores son correctas. g) Ninguna es correcta.	e) Todas las anteriores son correctas. f) Ninguna de las anteriores son ciertas.
11.- El procesamiento de una interrupción: a) Es llevada a cabo sólo por el Sistema Operativo, sin intervención del Hardware. b) Es llevada a cabo sólo por el Sistema Operativo, con intervención del Hardware. c) Obliga a guardar toda la información de estado del proceso interrumpido en el Stack del sistema. d) Devuelve siempre el control al proceso interrumpido. e) Todas las anteriores son correctas. f) Ninguna es correcta.	**12.- Las características del funcionamiento del procesador en modo usuario son:** a) Ejecutar las Instrucciones Privilegiadas b) Ejecutar un programa usuario. c) Puede ser interrumpido. d) Acceso restringido a recursos e instrucciones del procesador. e) Todas las anteriores son correctas. f) Ninguna de las anteriores son ciertas.
13.- Las instrucciones usuario se usan para las siguientes casos: a) Instrucciones de Entrada - Salida; b) Instrucciones para modificar registros de administración de memoria y cronómetro. c) Instrucciones HALT (parada). d) Instrucciones para activar y desactivar el sistema de interrupciones. e) Instrucciones para cambiar de modo usuario a privilegiado. f) Todas las anteriores. g) Ninguna de las anteriores.	**14.- Cuando ocurre una interrupción el PC (Program Counter) es cargado con un nuevo valor. Marque el valor que pueda ser cargado dentro del PC en condiciones normales.** a) Dirección de la próxima instrucción en memoria. b) Dirección de la primera instrucción de la Rutina de Servicios de la Interrupción c) Dirección de la primera instrucción del Programa del usuario. d) Dirección de la BIOS e) Todas las respuestas anteriores son correctas. f) Ninguna de las respuestas anteriores es correcta.
15.- Indicar qué situaciones son <u>incorrectas</u>, dado los siguientes datos: a) En un sistema de Uniprogramación pueden existir: Ninguno o un usuario, un proceso, un hilo, un CPU, un SO. b) En un sistema de Multiprogramación pueden existir: Ninguno, uno o más usuarios, uno o más procesos, uno o más hilos kernel, una CPU, un SO c) En un sistema de Multiprocesamiento pueden existir: Ninguno, uno o más de un usuario, múltiples procesos, múltiples hilos kernel por proceso, una CPU, un SO. d) En un sistema Distribuido pueden existir: Ninguno, uno o más usuarios, múltiples procesos, múltiples CPU's, múltiples SO	**16.- Indique que concepto definen a una maquina extendida...** a) Maquina virtual. b) El conjunto de S.O. + el hardware. c) Idem B) + el usuario. d) La maquina convencional + el S.O. e) Un conjunto de instrucciones y facilidades que no están presentes en el nivel de maquina convencional. f) Un conjunto de entidades activas que definen una estructura de hardware que actúa como interfase con el usuario. g) Todas las respuestas anteriores son correctas. h) Ninguna de las respuestas anteriores es correcta.
17.- Cual o cuales de las siguientes afirmaciones son correctas desde el punto de vista del S.O.. a) Implementa una política debido a las prioridades. b) Implementa una protección debido a la seguridad. c) Implementa una estrategia generando un orden. d) Implementa una autoridad debido a las decisiones. e) Implementa una contabilidad debido al control. f) Implementa una demanda debido a la competencia. g) Implementa una optimización debido a la contabilidad. h) Todas las respuestas anteriores son correctas. i) Ninguna de las respuestas anteriores es correcta.	**18.- Una maquina virtual es.....** a) Una interfase hombre/maquina que permite a este utilizar una memoria de grandes dimensiones, disponiendo de espacio en memoria secundaria. b) La simulación vía software de una computadora de dimensiones mayor a la real. c) Idem b) pero haciendo uso de recursos de hardware y software al mismo tiempo. d) Una interfase usuario-computador que define desde el punto de vista del usuario una maquina con características especiales que hacen transparente los procesos reales producidos por el hardware. e) Un ambiente de software. f) Un estado ideal desde el punto de vista de las computaciones donde el usuario define las mismas en forma independiente a como estas se realizan en realidad. g) Todas las anteriores. h) Ninguna de las anteriores.
19.- Una Arquitectura computacional se caracteriza por.... a) El Orgware definido para su estructura. b) El Firmware de los Circutos combinacionales. c) El Software que ejecuta. d) El Hardware definido para n bits de procesamiento. e) El Peopleware de su estructura interna. f) Todas las anteriores son correctas g) Ninguna es correcta.	**20.- Los objetivos para diseñar un S.O son....** a) Difícil de usar. b) Complejo en las interfases. c) Portable entre maquinas. d) Escalable. e) Confiable. f) Todas las anteriores. h) Ninguna de las anteriores.

Respuestas a las preguntas

1- La arquitectura por capas (layers) de Dijsktra determina que la capa (n-1) le abstrae ciertas funciones a la capa n , mientras que la capa (n+1) le entrega servicios a esta capa n . ¿Es verdad esto?. Justifique.

Falso. La capa n-1 abstrae las funciones y brinda servicios a la capa n.

2- ¿Es verdad que en cada ciclo de instrucción el sistema operativo se fija si aconteció alguna interrupción? Justifique.

Falso. El que pregunta si hay una interrupción es el hardware (First Level Interrupt Handler) y no el sistema operativo pues él no está ejecutándose en ese momento.

3- Nombre las principales ventajas y desventajas de diseñar un sistema operativo usando una arquitectura de maquinas virtuales. ¿Cuales son las ventajas y desventajas para el usuario?

Ventajas:

a) Cada usuario del sistema puede usar un sistema operativo distinto.
b) Permite un alto nivel de protección. Todas las maquinas virtuales son independientes. Si bien algunos recursos del computador se comparten para crear maquinas virtuales, dichos recursos no se comparten directamente. Para lograr esto existen dos estrategias :
   - Minidisco: Sigue el modelo de un disco físico compartido, pero se pone en practica mediante un software (con esta técnica los archivos se pueden compartir).
   - Red de maquinas virtuales: Cada una puede enviar información a través de una red virtual de comunicaciones. Esta red tiene como modelo a la red física de comunicaciones, pero se implementa en software.
c) Son especialmente útiles para el diseño y desarrollo de nuevos sistemas operativos.

Desventajas
a) Difícil implementaron. Se necesita un duplicado exacto del hardware subyacente.
b) La simulación del hardware es muy costosa en recursos.
c) Cada maquina virtual es mas lenta que la maquina real, ya que la CPU se multiprograma entre varias maquinas virtuales , lo que las hace mas lentas.

4) ¿A qué áreas provee servicios un S.O.?

Brevemente un sistema operativo brinda normalmente servicios en las siguientes áreas:

Desarrollo de programas: El S.O. ofrece una variedad de facilidades y servicios tales como los editores y los debuggers, para ayudar al programador en la creación de programas. Normalmente, estos servicios están en forma de programas de utilidad que no forman parte del S.O. pero que son accesibles a través de este.

Ejecución de programas: Las instrucciones y los datos se deben cargar en la memoria principal, los archivos y los dispositivos de E/S se deben preparar otros recursos. El S.O. administra todas estas operaciones por el usuario.

Acceso a dispositivos de E/S: Cada unos de los dispositivos requiere su propio y peculiar conjunto de instrucciones o señales de control para su funcionamiento. El sistema operativo provee una interfase uniforme que esconde estos detalles de tal manera que el programador pueda acceder a estos dispositivos usando simples lecturas (READ) y escrituras (WRITE).

Control de acceso a archivos: En el caso de los archivos, el control debe incluir una compresión, no solo de la naturaleza del dispositivo de E/S sino el formato de los archivos y el medio de

almacenamiento. Además, en el caso de un sistema con múltiples usuarios, provee mecanismos de protección para controlar el acceso a archivos.

Acceso al sistema: En el caso de un sistema compartido o público, el S.O. controla el acceso al sistema como un todo y a los recursos específicos del sistema. Las funciones de acceso deben proveer protección de los recursos y datos de los usuarios no autorizados.

Detección de errores y respuesta: Entre estos se incluyen los errores internos y externos del hardware, tales como los errores de memoria, fallos o mal funcionamiento de dispositivos y distintos tipos de errores de software, como desbordamiento aritmético, intento de acceder a una zona de memoria no permitida y la incapacidad del S.O. para satisfacer a una aplicación. El S.O. debe dar una respuesta que elimine la condición de error con el menor impacto posible sobre las aplicaciones que están en ejecución.

Contabilidad: Un buen S.O. debe recoger estadísticas de utilización de los diversos recursos y supervisar los parámetros de rendimiento tales como el tiempo de respuesta.

5) ¿Qué protección de la información y seguridad brinda un S.O.?

En general los problemas que se controlan es el de acceso a la computadora y la información almacenada en ella.

La protección y seguridad que brinda un sistema operativo las podemos agrupar dentro de estas tres categorías:

Control de acceso: Regulación del acceso del usuario al sistema completo, a los subsistemas y a los datos, así como a regular el acceso de los procesos a los recursos y objetos del sistema.

Control del flujo de información: Regula el flujo de datos dentro del sistema y su distribución a los usuarios.

Certificación: Es relativa a la demostración de que el acceso y los mecanismos de control del flujo se llevan a cabo de acuerdo a las especificaciones

6. ¿Cuáles son las funciones de un Sistema Operativo?

Un sistema operativo posee cuatro funciones básicas:

1. **Inicialización del sistema:** arranque en frío (Booteo o IPL) y arranque en caliente.

2. Interf**az usuario/computadora o máquina extendida**: El programa de sistemas más importante es el Sistema Operativo, que oculta al programador los detalles del hardware y le proporciona una interfaz cómoda para utilizar el sistema. el usuario final ve al sistema informático en término abstracto y seguro

3. **Administración de recursos:** El Sistema Operativo es el responsable de la gestión de estos recursos de un computador. La asignación del recurso es controlada por el Sistema Operativo y por el hardware de gestión de memoria. El Sistema Operativo decide cuándo puede utilizarse un dispositivo de E/S, controla el acceso y la utilización de los archivos, y determina cuánto tiempo del procesador debe dedicarse a la ejecución de cada programa.

4. **La Protección:** Su función es mantener la integridad de los recursos y de los procesos. Para ello se ocupa del control:

- **de acceso**: autorizando el acceso de usuarios legítimos al sistema completo, a los subsistemas y a los datos, así como también el acceso de los procesos autorizados a los recursos y objetos del sistema.

- **del flujo de información**: Regula el flujo de datos dentro del sistema y su distribución a los usuarios.

- **De Autenticación**: Es relativa a la demostración de que el acceso y los mecanismos de control del flujo se llevan a cabo de acuerdo a las especificaciones.

7.- Las Instrucciones No Enmascarables pueden interrumpir cualquier tipo de instrucción (inclusive las de tipo Kernel). En cambio, las Enmascarables deben esperar a que la instrucción en curso finalice antes de ser atendida.

8.- Los métodos básicos que utiliza el S.O. para brindar los servicios son System Programs y System Calls

Respuestas del múltiple choice.

1.- b, c, e.	2.- a, c, d.	3.- b, e.	4.- a, b, c, d.	5.- b, c, d, e.
6.- f.	7.- a, c, d, e.	8.- a, b, d.	9.- b, c.	10.- a, c, e
11.- b, c.	12.- b, c, d.	13.- g.	14.- b.	15.- c.
16.- a, b.	17.- a, b, c, d, e.	18.- d, e, f.	19.- a, c, d.	20.- c, d, e.