Clase 18 - Sistemas Operativos

IIC1001 - Algoritmos y Sistemas Computacionales

Cristian Ruz – cruz@uc.cl

Martes 12-Mayo-2026

Departamento de Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica de Chile

Contenidos

Contacto

Temas

Sistemas Operativos

Contenidos

Contacto

Temas

Sistemas Operativos

Contacto

🐐

vicente.cabra@uc.cl

Vicente Cabra

Ayudante jefe



🐸

alejandro.tapia@uc.cl

Alejandro Tapia

Ayudante jefe



🛼

gabriel.sandoval@uc.cl

Gabriel Sandoval

Ayudante jefe



Contenidos

Contacto

Temas

Sistemas Operativos

Temas del curso

Sistemas computacionales

- Representación datos, números y codificación
- Funcionamiento hardware, procesadores y memoria.
- Funcionamiento de sistemas operativos
- Funcionamiento de Internet
- Herramientas computacionales: github + latex

Algoritmos

- Algoritmos y resolución de problemas
- Eficiencia algorítmica
- Estructuras secuenciales y ordenamiento
- Grafos y árboles

Próximos eventos

Jueves 14-Mayo: Taller Git

- Individual

Martes 19-Mayo: Receso

- No hay clases

Martes 26-Mayo: Interrogación 2

- 17:30

Contenidos

Contacto

Temas

Sistemas Operativos

¿Qué tenemos?

¿Qué tenemos?

- Hardware ...y un set de instrucciones

¿Y hacia dónde vamos?

- A ejecutar un programa

¿Cómo usar el computador?

Necesitamos ser capaces de usar:

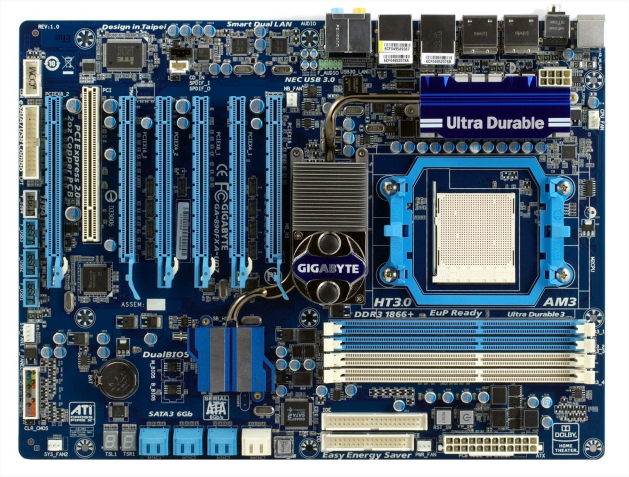
CPU
Para ejecutar programas

Memoria
Para almacenar y leer programas

I/O
Para interactuar con el ambiente

¿Cómo usar el computador?

Pero tenemos:



Para usarla debemos escribir cada instrucción



¿En serio?

Antes de ejecutar:

```
movl    $2, -8(%rbp)
movl    $3, -12(%rbp)
movl    -8(%rbp), %eax
addl    -12(%rbp), %eax
movl    %eax, -16(%rbp)
movl    -16(%rbp), %esi
movb    $0, %al
callq   _printf
```

Necesitamos decir al computador:

- Qué hacer cuando se apreta una tecla, o se mueve el mouse.
- Cómo desplegar un caracter o imagen en pantalla
- En qué posición de la memoria podemos escribir
- Cómo leer archivos desde el disco

MUCHOS programas requieren estos mismos servicios

¿Y ahora quién podrá ayudarnos ...?

Un programa “maestro” que se ejecuta permanentemente y permite:

- Entender entradas de teclado, mouse
- Desplegar caracteres, imágenes en pantalla
- Leer y escribir desde ciertos lugares de memoria
- ...y en suma ...**ejecutar programas**

Necesitamos un

SISTEMA OPERATIVO

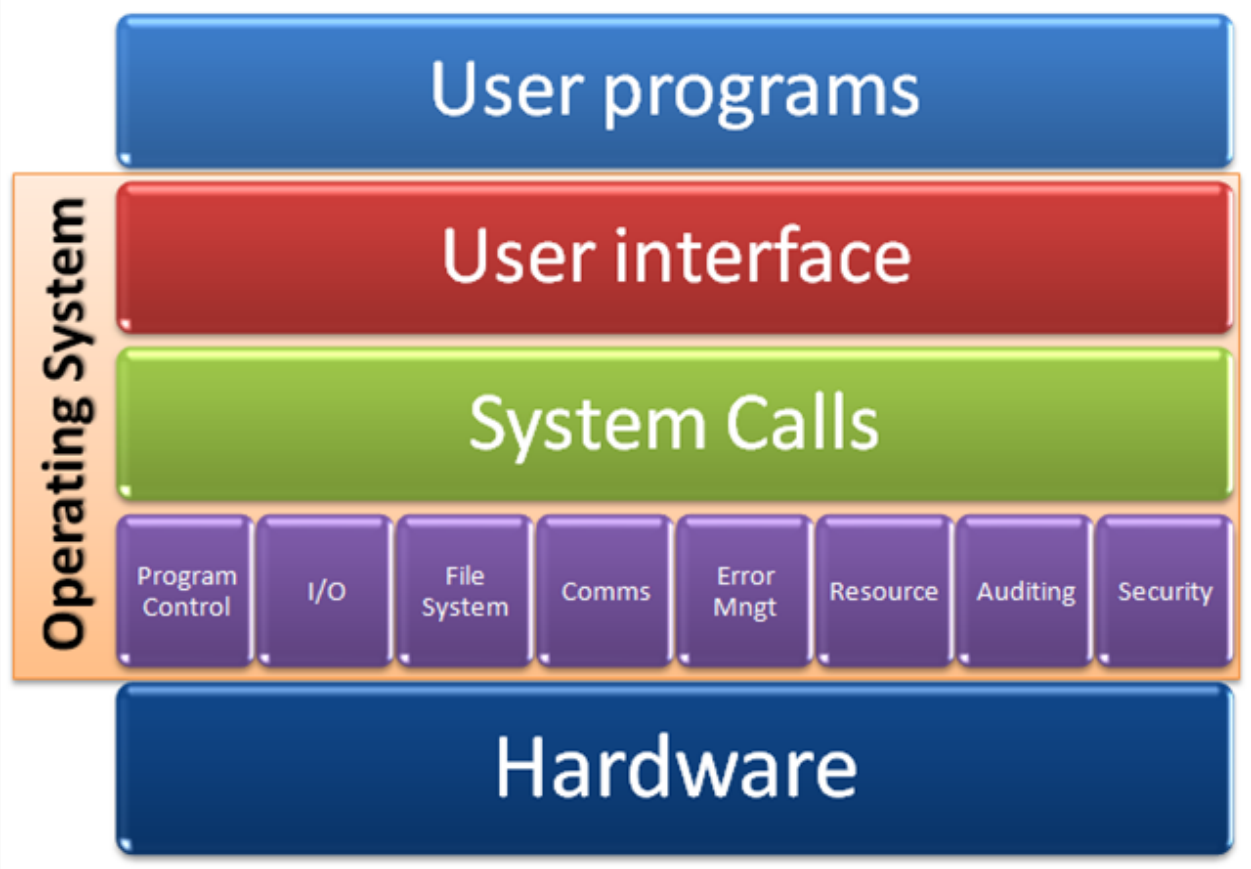
Entonces ...¿qué es un sistema operativo?

Programa que...

- Permite ejecutar programas (¿“procesos”?) sobre la CPU
- Administrar la memoria disponible entre uno o más procesos
- Administrar dispositivos de I/O para que puedan ser usados por los procesos

Permite utilizar el computador

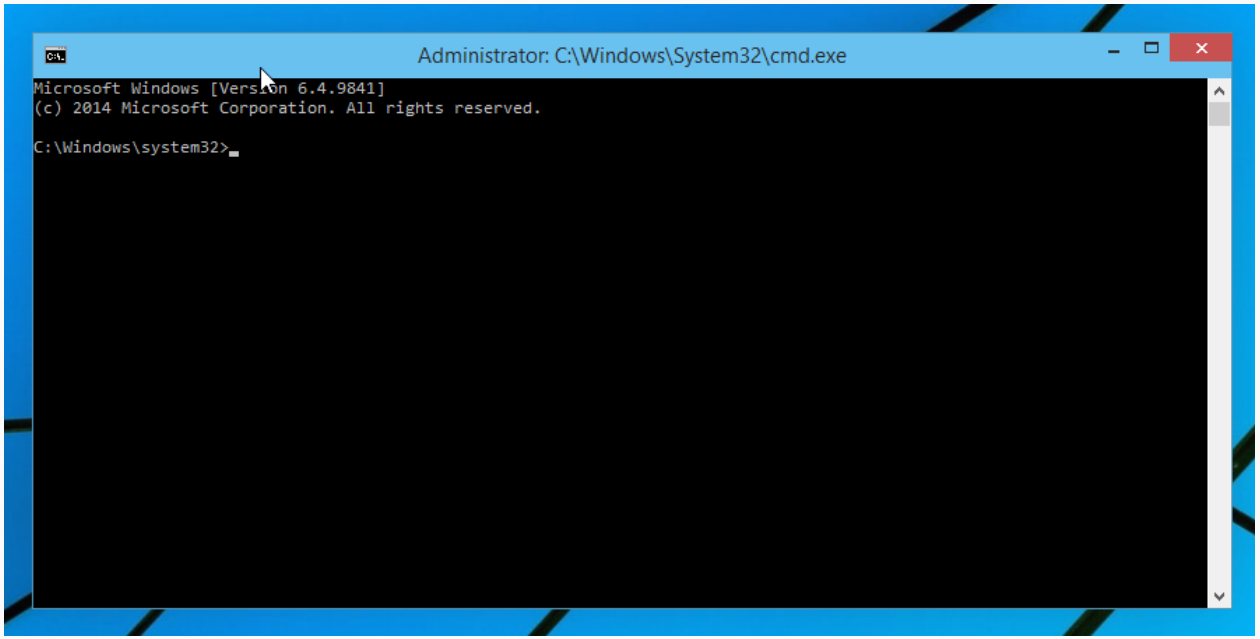
¿Qué es un sistema operativo?



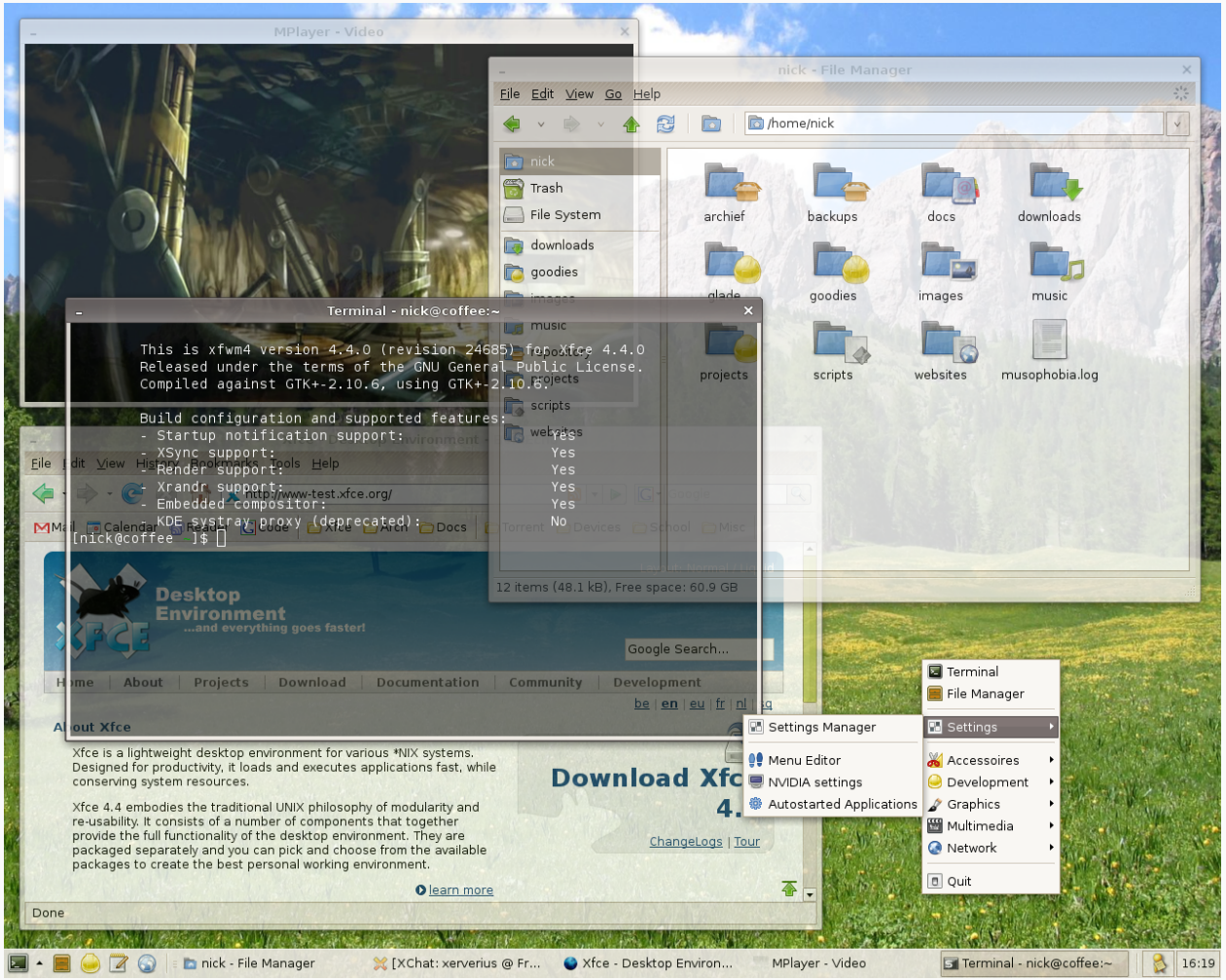
Interfaces de Usuario (UI)

También conocidos como **shells**

```
cruz@hercules:~$ cd iic2333/
cruz@hercules:iic2333$ ls
examples
cruz@hercules:iic2333$ cd examples/
cruz@hercules:examples$ ls
fork2.cpp  fork3.c  fork.c  shmem-cons.c  shmem-prod.c  waitZombies.c
cruz@hercules:examples$
```

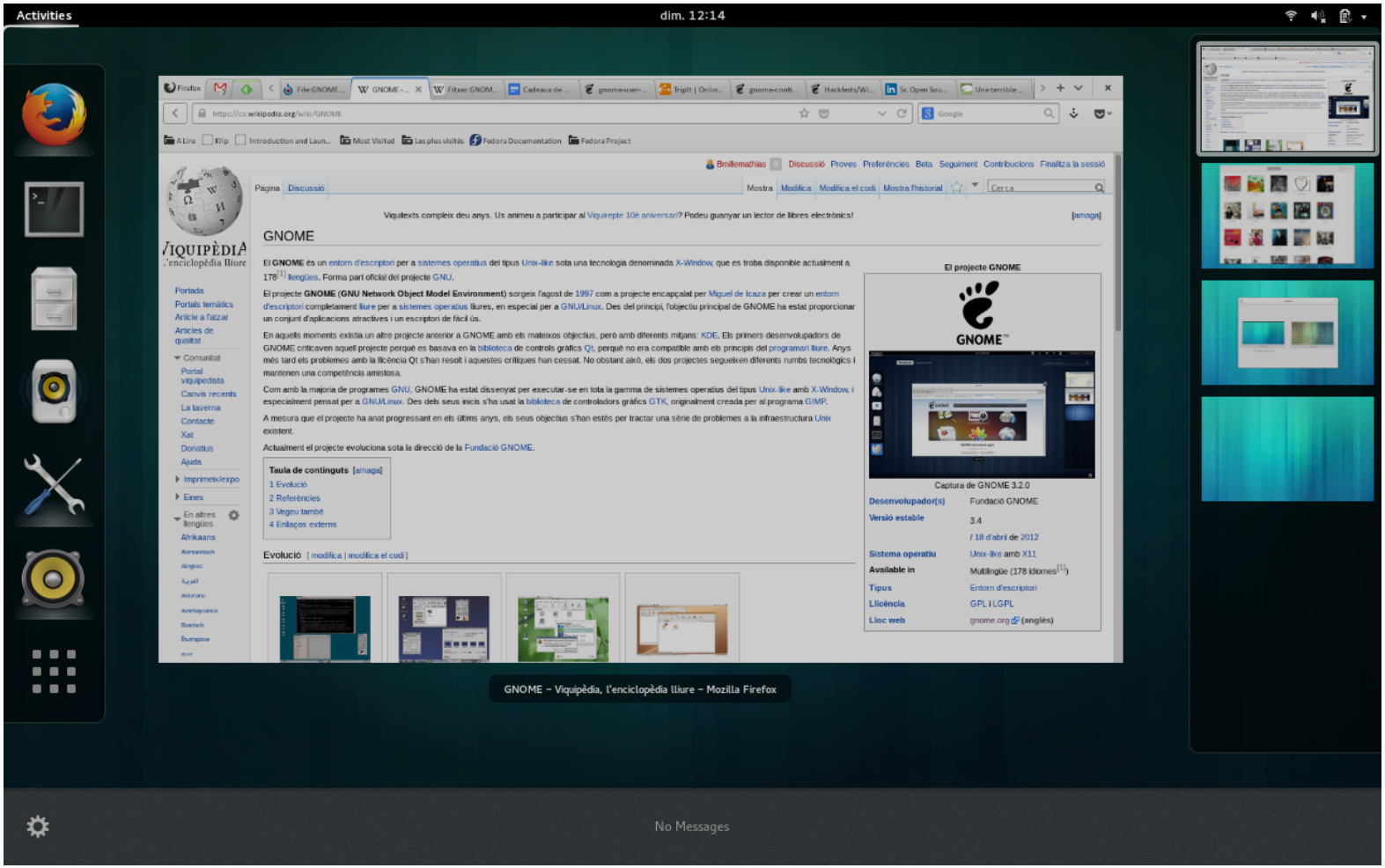


Interfaces de Usuario (UI)



Xfce

Interfaces de Usuario (UI)



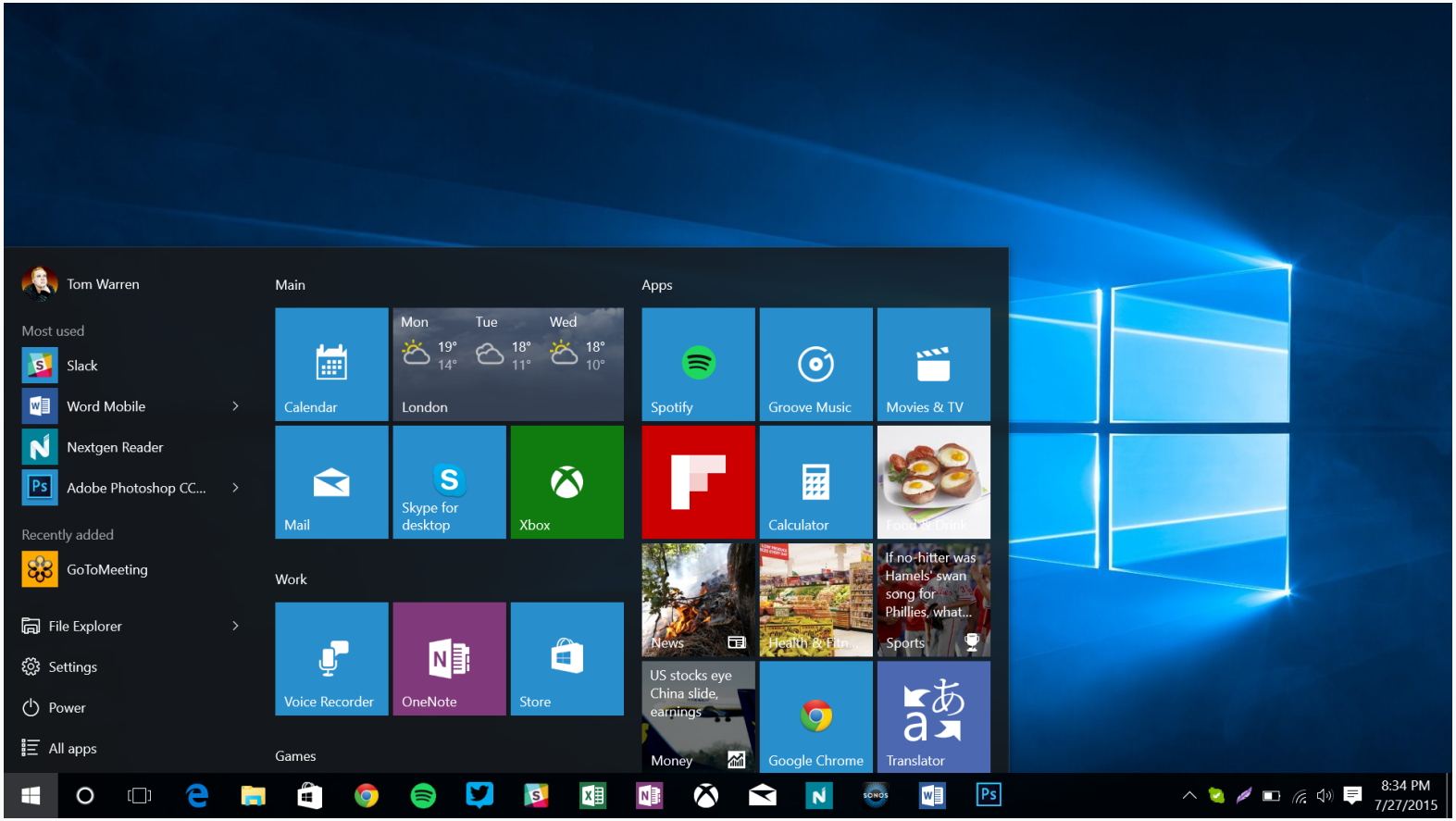
GNOME

Interfaces de Usuario (UI)



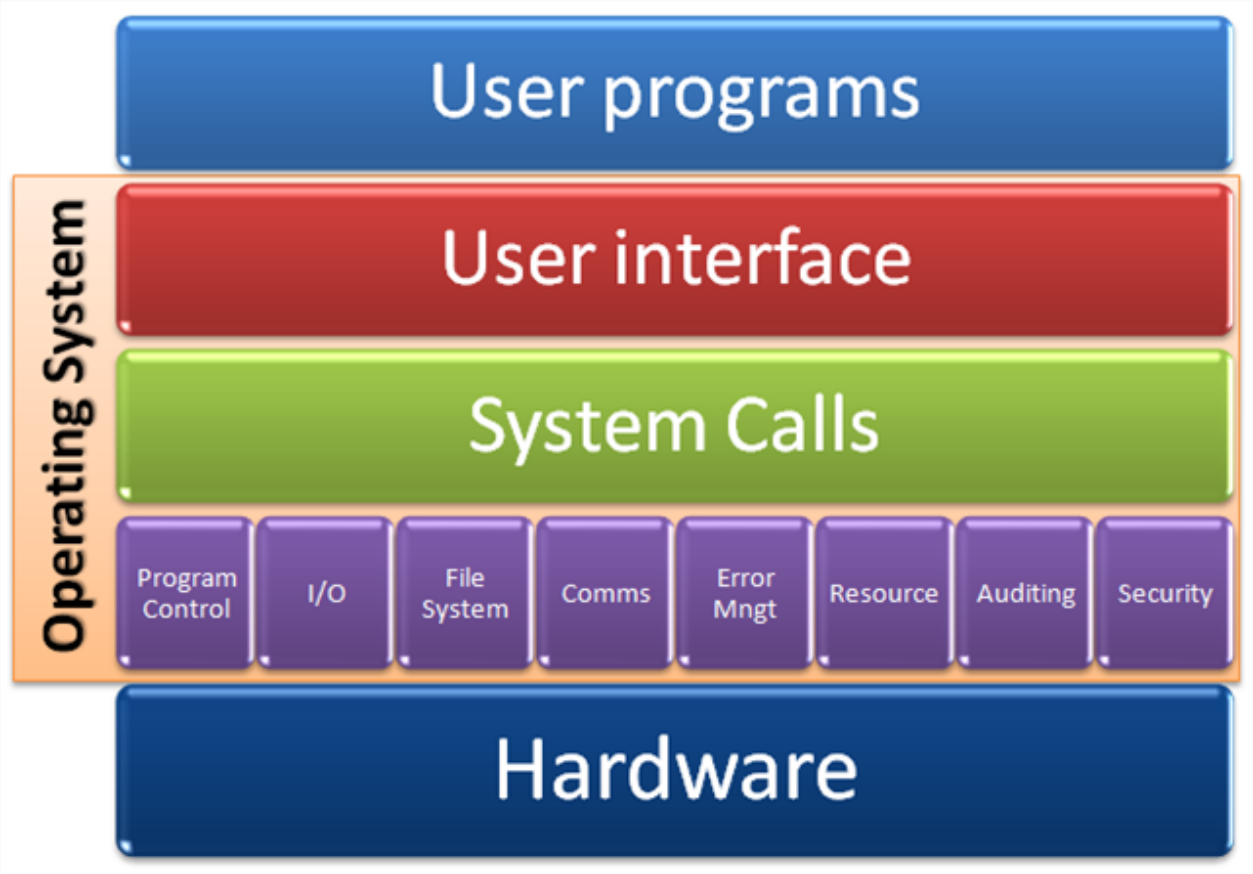
GNOME

Interfaces de Usuario (UI)



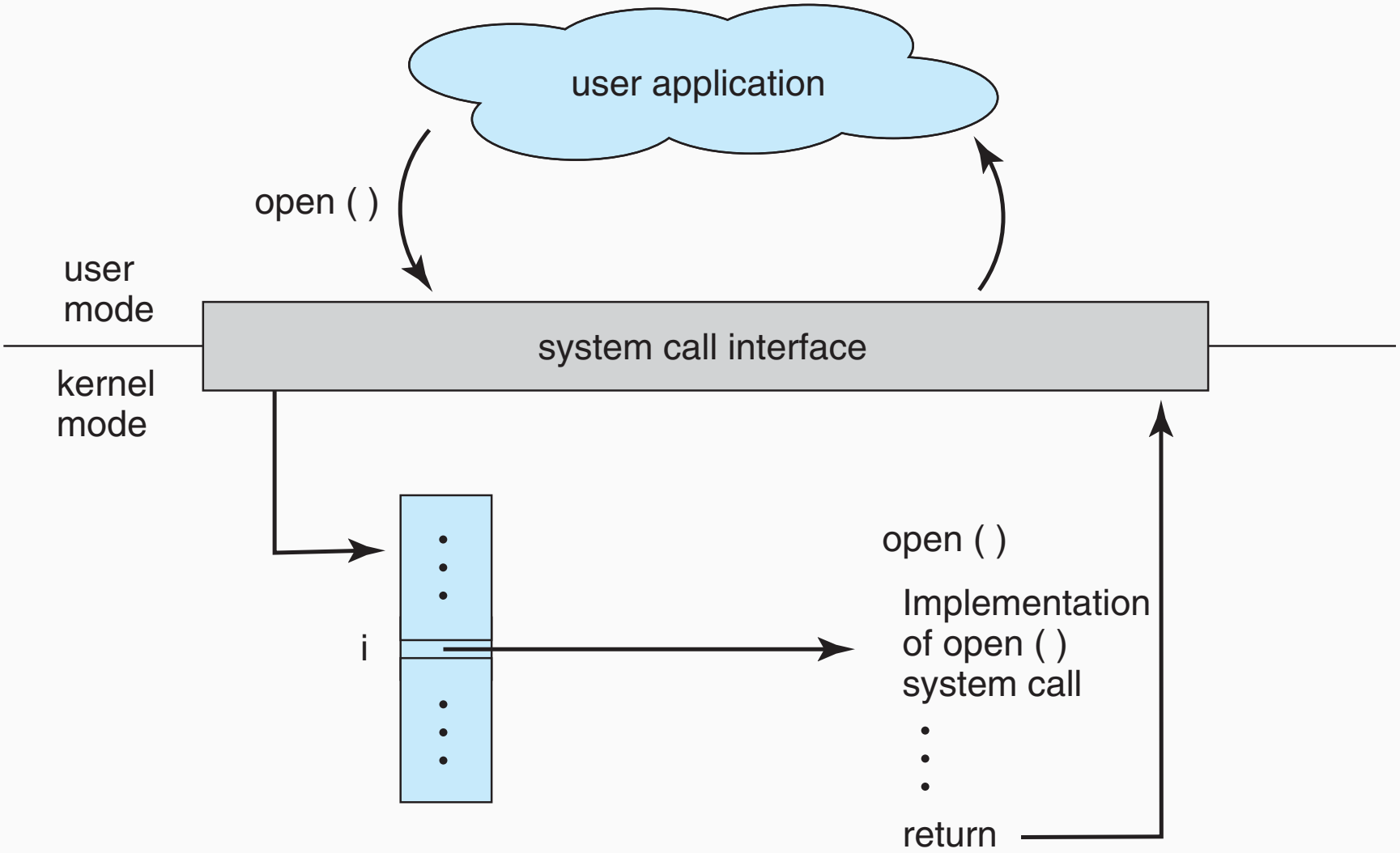
GNOME

¿Qué es un sistema operativo?



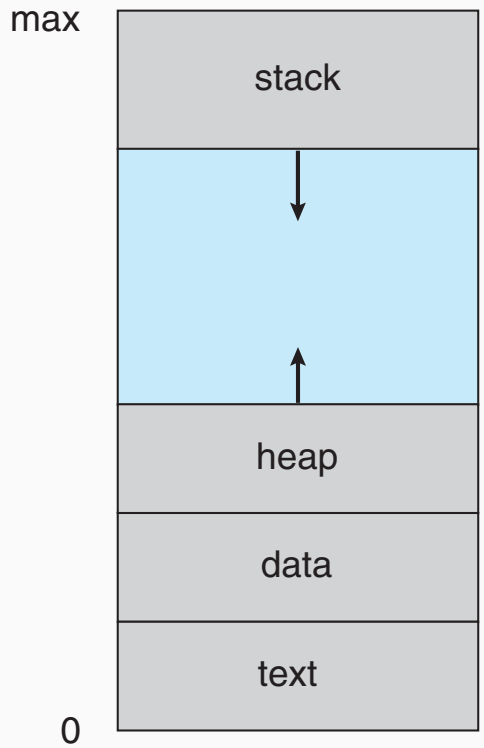
Llamadas al sistema

Programas utilizan llamadas al sistemas frecuentemente

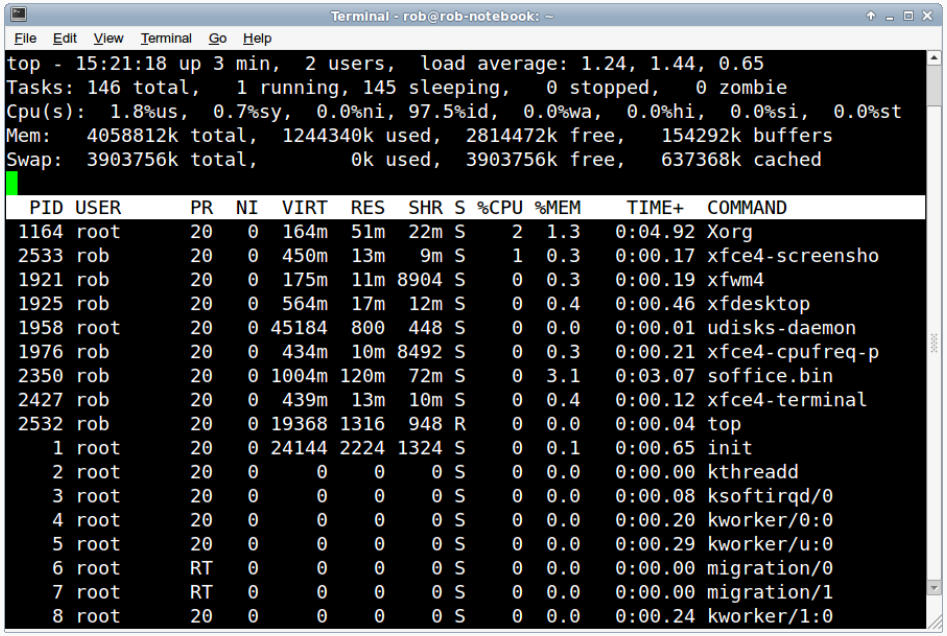


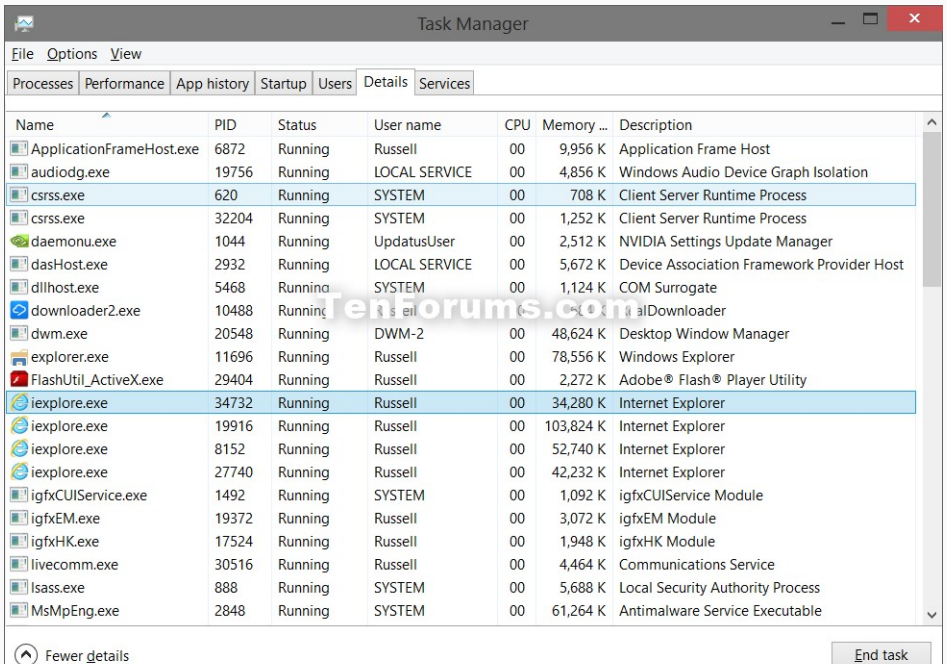
¿Qué hay en un proceso?

- **Código** ≡ texto ...información estática
- **Program Counter** + **registros**, ...actividad actual
- **Stack**, datos temporales
  - Parámetros de función, direcciones de retorno, variables locales
- **Datos**, variables globales
- **Heap**, memoria asignada dinámicamente (en tiempo de ejecución)



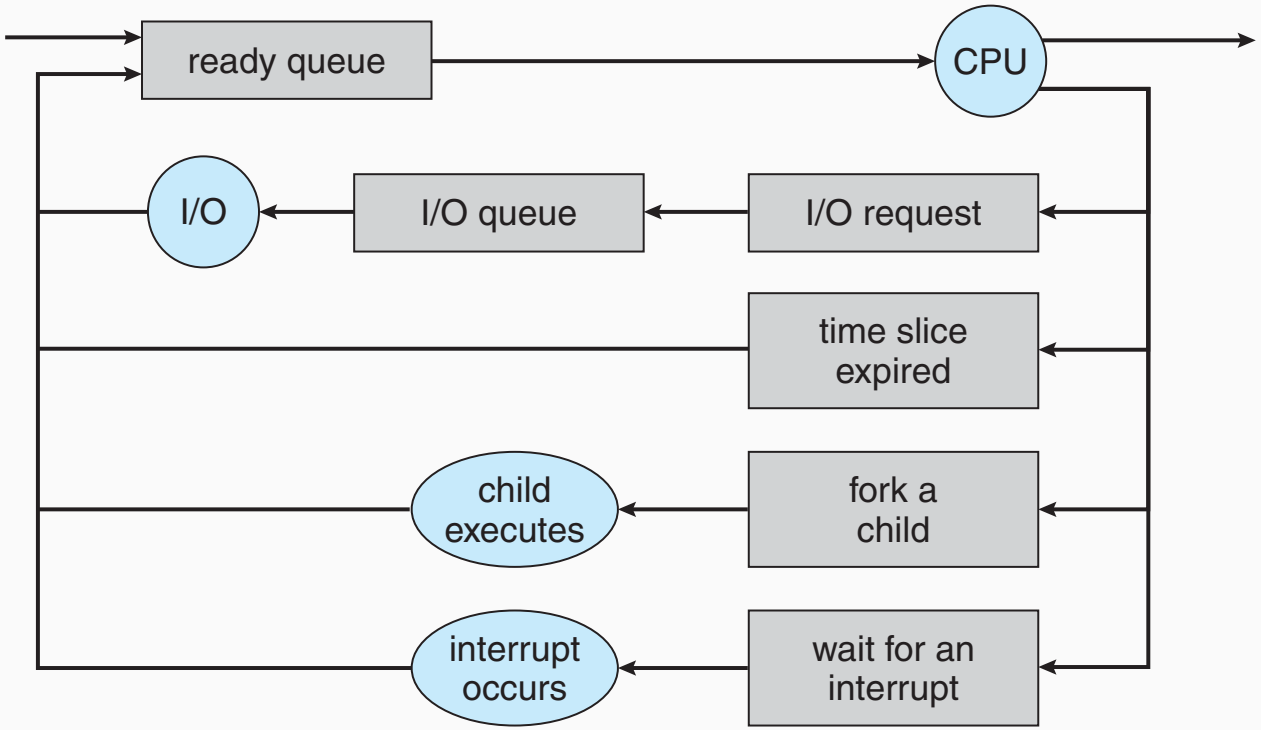
Muchos procesos quieren ocupar la CPU





Colas de *scheduling*

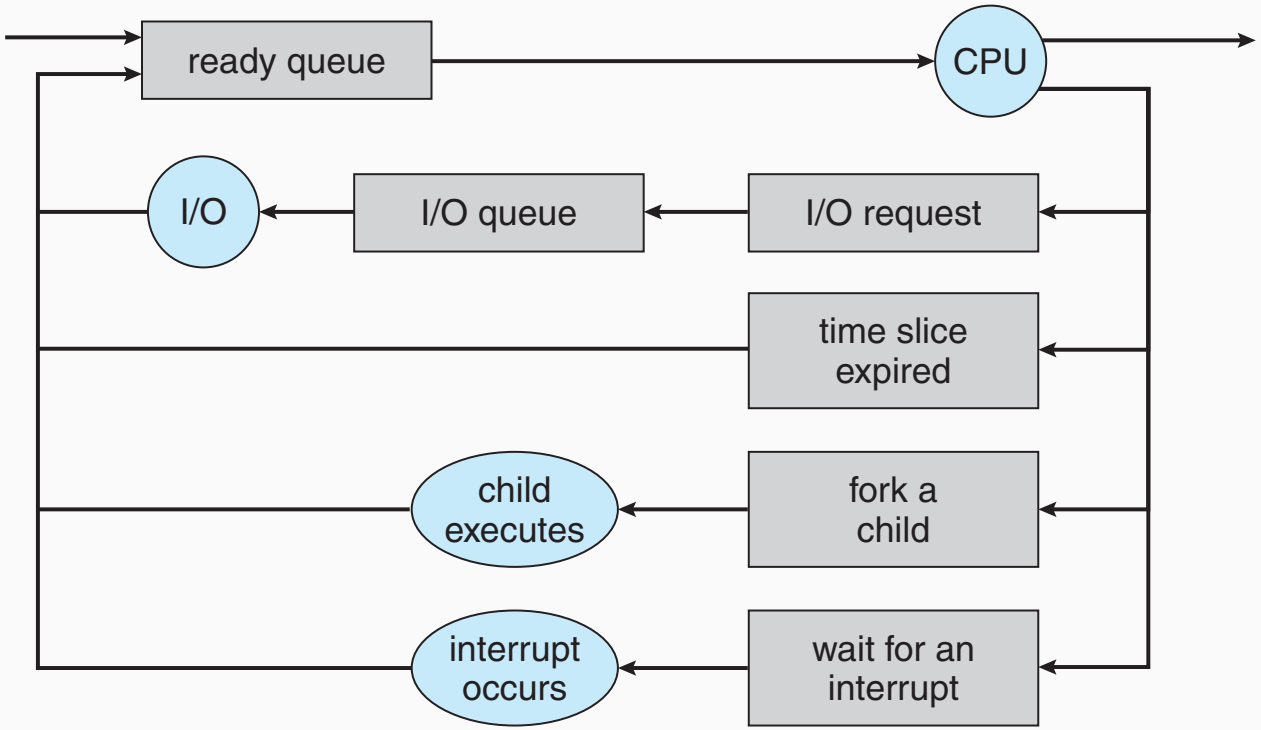
Procesos se *encolan* para ocupar la CPU



¿Bajo qué criterio?

Colas de *scheduling*

Procesos se *encolan* para ocupar la CPU



¿Bajo qué criterio? $\longrightarrow$ **algoritmos de *scheduling***

Memoria

- Un gran arreglo de *bytes*
- Cada uno con su propia **dirección**

- PC lee instrucciones desde una dirección de memoria
- Instrucciones pueden requerir leer operandos desde otra dirección de memoria
- Resultados suelen ser almacenados en alguna dirección de memoria

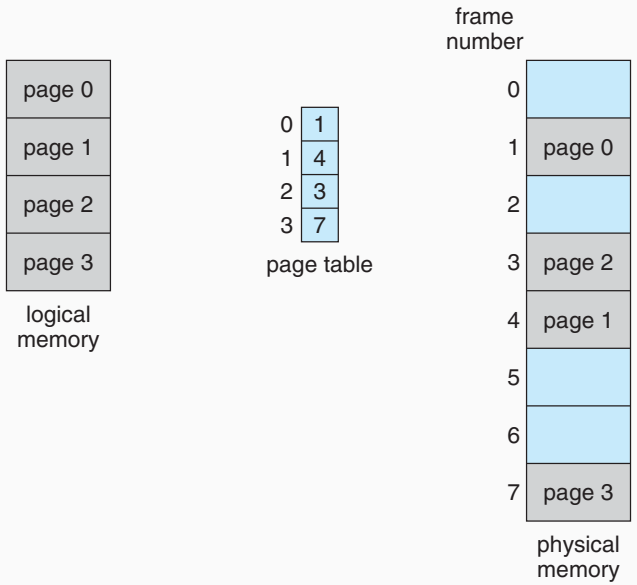
Paginación

División de memoria en unidades de tamaño fijo:

- Memoria física se divide en **frames** (marcos)
- Memoria lógica se divide en **páginas**

Dirección de memoria paginada contiene:

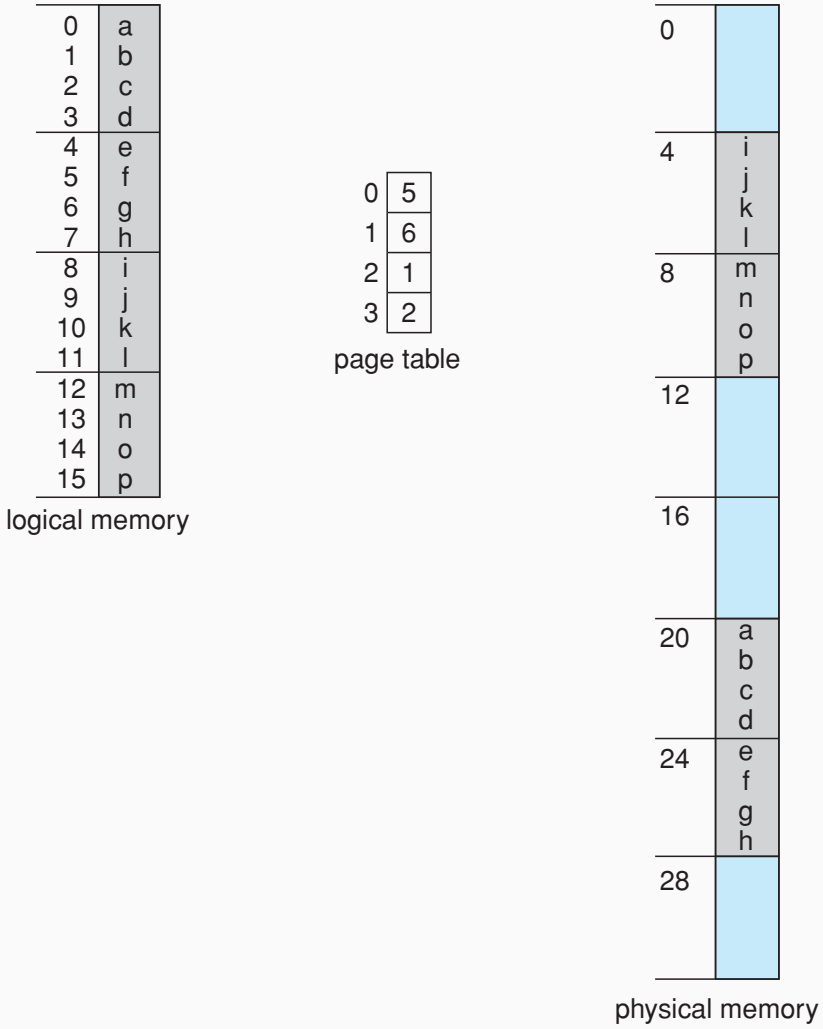
$$\langle \texttt{numeroDePagina}, \texttt{offset} \rangle$$



Ejemplo de paginación

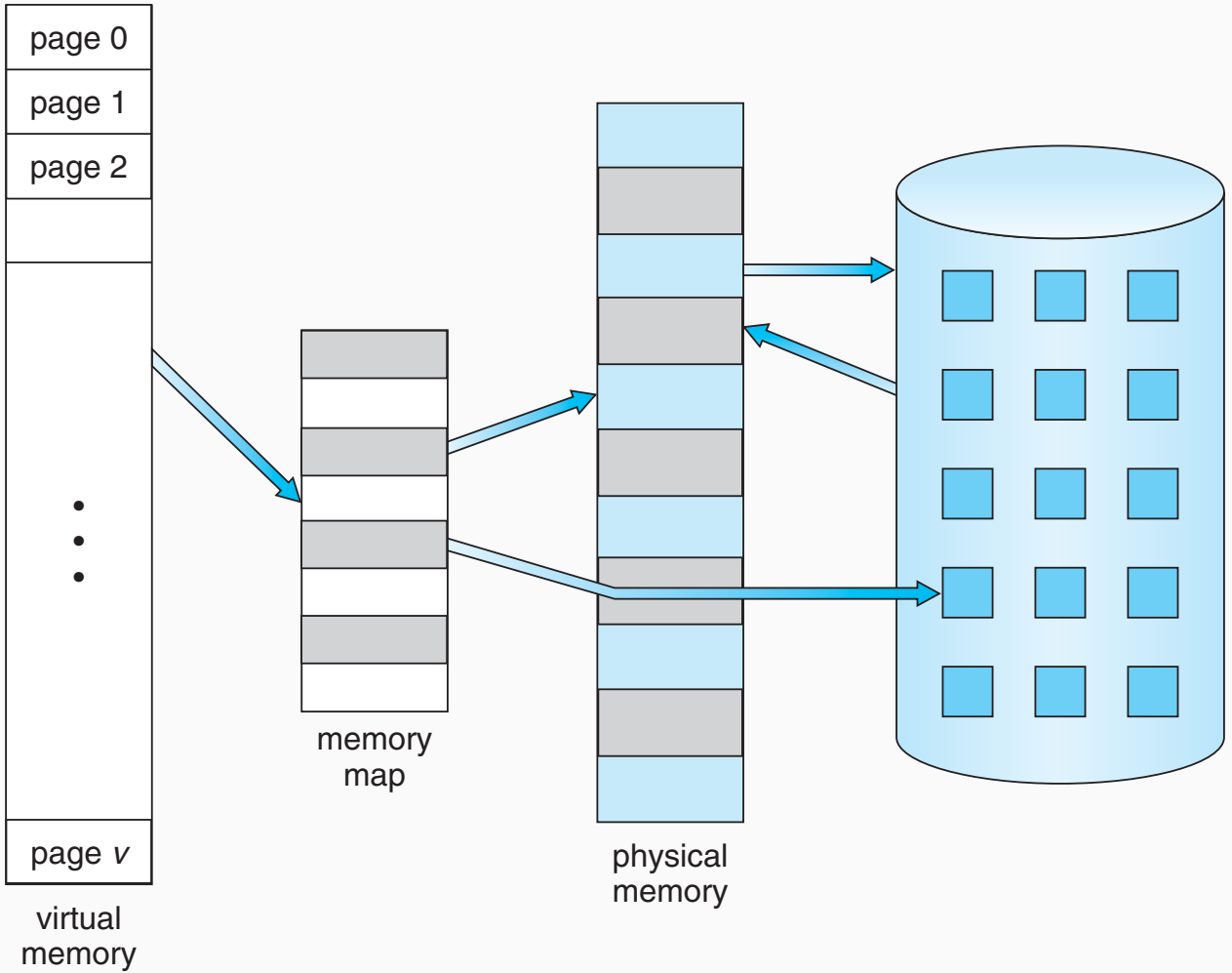
Páginas de 2^n bytes, $n = 2$. Direcciones lógicas de m-bit, $m = 4$.

Memoria física de 32 bytes == 8 páginas.



Memoria Virtual: Funcionamiento

Memoria Virtual aumenta la separación entre la memoria visible por el programador, y la memoria física de la máquina en que se ejecuta el programa.



Archivos

Archivo
Colección de información en almacenamiento secundario

- Unidad de almacenamiento para el usuario
- Agrupación lógica de *bytes*

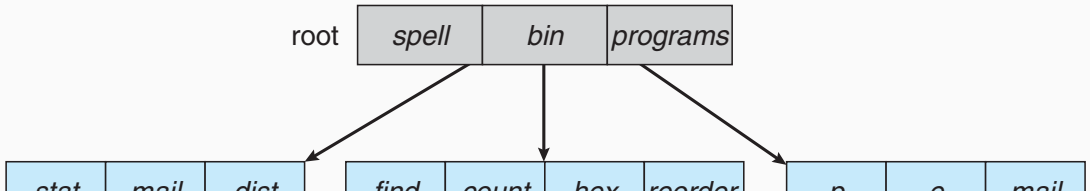
Contenido: cualquier cosa

- Significado está determinado por la forma en que se utiliza su contenido
- Texto, imagen, código fuente, código binario (ejecutable), ...

Directorios

¿Cómo encontrar archivos?

- Sistemas de archivos mantienen *listas* de archivos y sus ubicaciones en bloques de disco
- Archivos se listan en una **tabla de contenidos** ó **árbol de directorios**
- Camino desde la raíz al directorio: **ruta** ó **path**
- Cada proceso tiene un directorio actual: **current directory** (`pwd`)
- **Ruta absoluta**: Ruta desde la raíz. Ej: `/spell/mail/exp`
- **Ruta relativa**: Ruta desde la ubicación actual (*current directory*). Ej: `p/list` solo existe si el directorio actual es `/programs/`



Sistemas de archivos

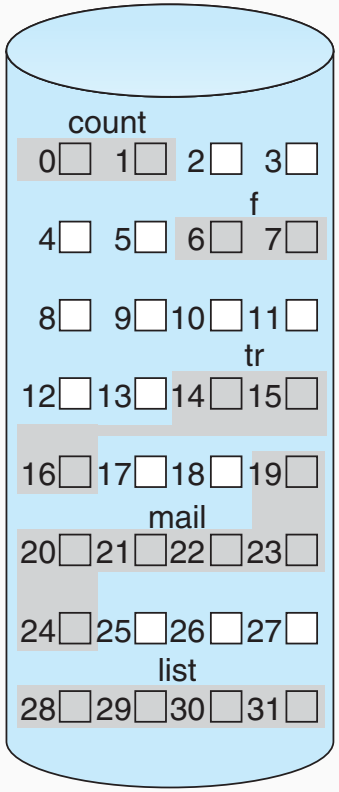
Un **sistema de archivos** permite obtener una dirección de disco, a partir de un datos simbólicos: nombre de archivo, directorio, rutas, links, ...

Algunos ejemplos:

- ISO9660 (CD-ROMs)
- UFS: Unix File System, basado Berkeley Fast File System (FFS)
- FAT, FAT32, NTFS: Windows
- ext3, ext4: Linux
- GFS, FUSE

Asignación de bloques

- Archivo de n bloques, bloque inicial b
- Bloques: $b, b+1, \ldots b+n-1$
- Fácil de acceder a siguiente bloque

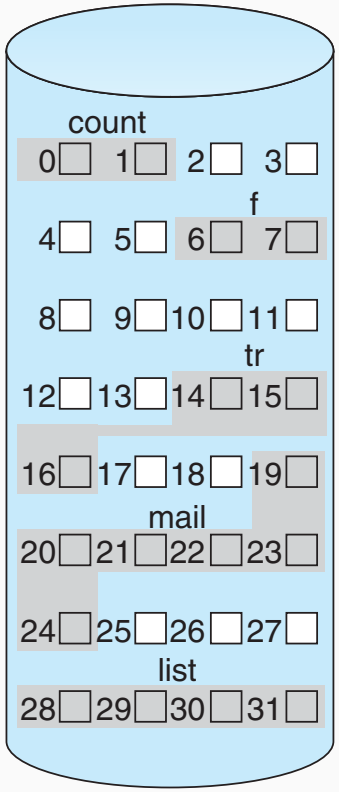


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Asignación de bloques

- Archivo de n bloques, bloque inicial b
- Bloques: $b, b+1, \ldots b+n-1$
- Fácil de acceder a siguiente bloque
- -Problema: fragmentación externa

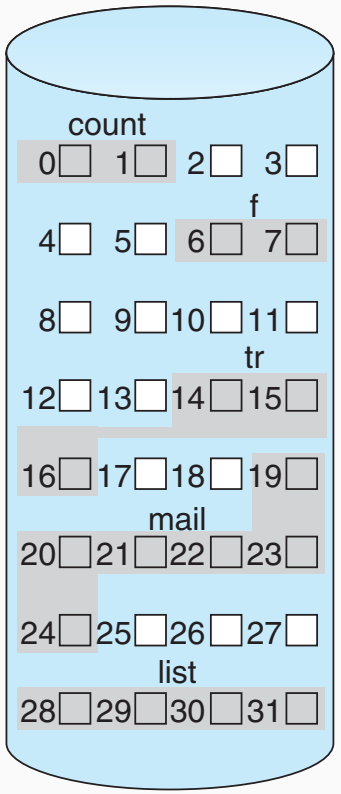


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Asignación de bloques

- Archivo de n bloques, bloque inicial b
- Bloques: $b, b+1, \ldots b+n-1$
- Fácil de acceder a siguiente bloque
- -Problema: fragmentación externa
- -Require operaciones de **compactación** (defragmentación)

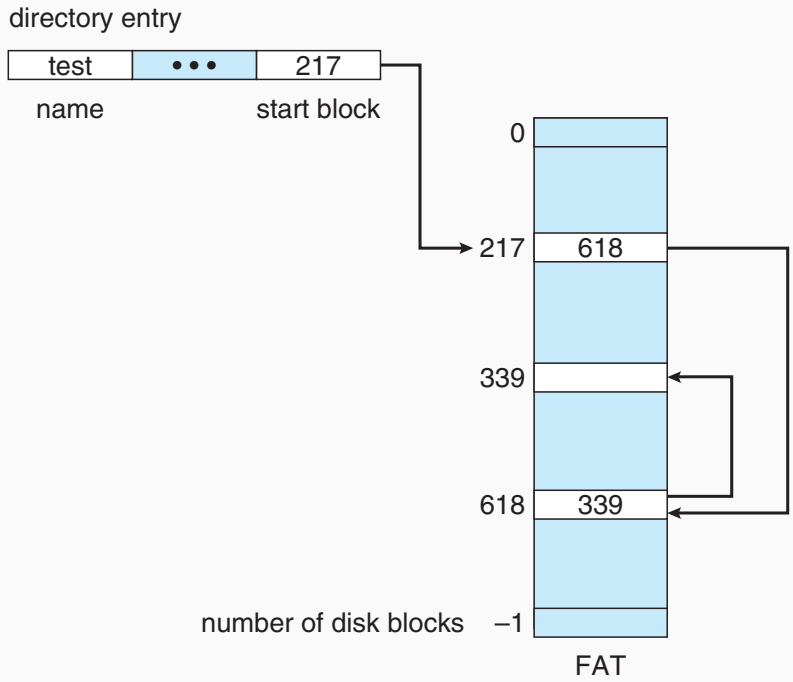


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

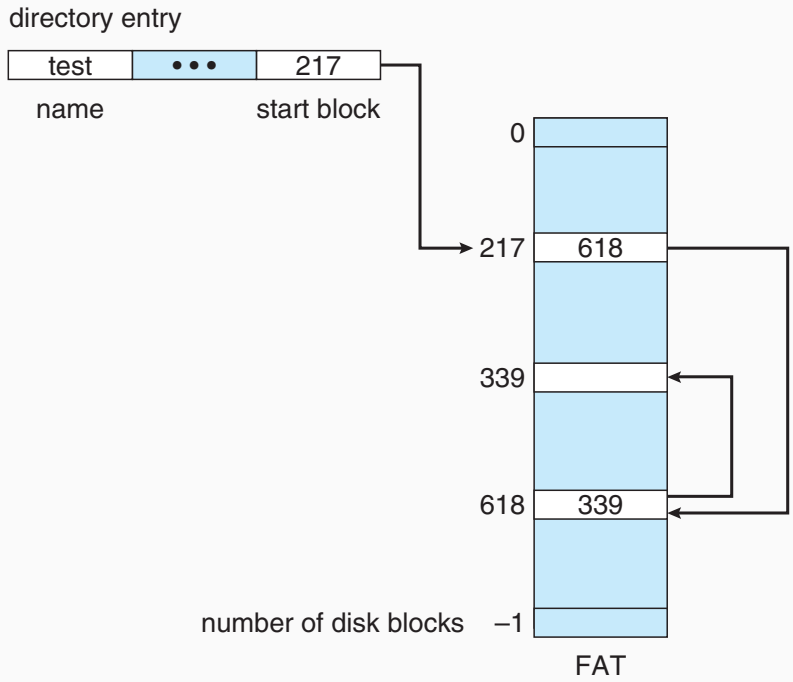
Asignación de bloques

- FAT: **File Allocation Table**
- Tabla al inicio del disco con una entrada por archivo
- Último cluster almacena EOF (*end-of-file*)



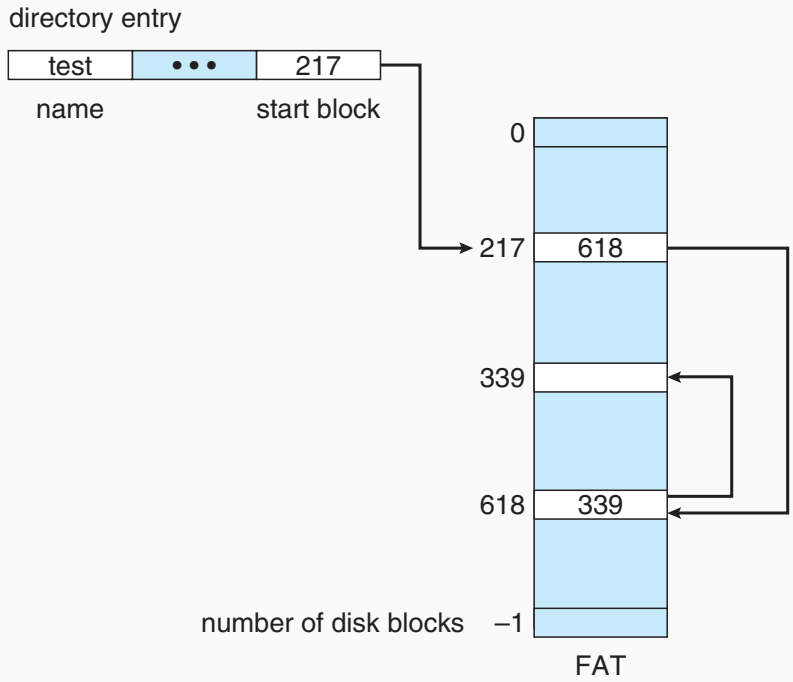
Asignación de bloques

- FAT: **File Allocation Table**
- Tabla al inicio del disco con una entrada por archivo
- Último cluster almacena EOF (*end-of-file*)
- +Mejor acceso aleatorio



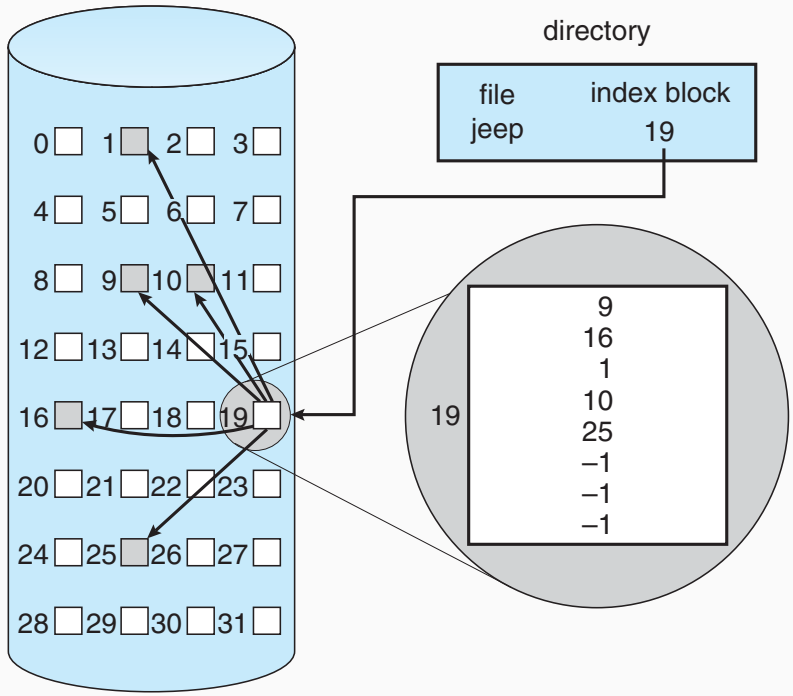
Asignación de bloques

- FAT: **File Allocation Table**
- Tabla al inicio del disco con una entrada por archivo
- Último cluster almacena EOF (*end-of-file*)
- +Mejor acceso aleatorio
- -Limitado por tamaño de tabla (para FAT32, 4GB por archivo)



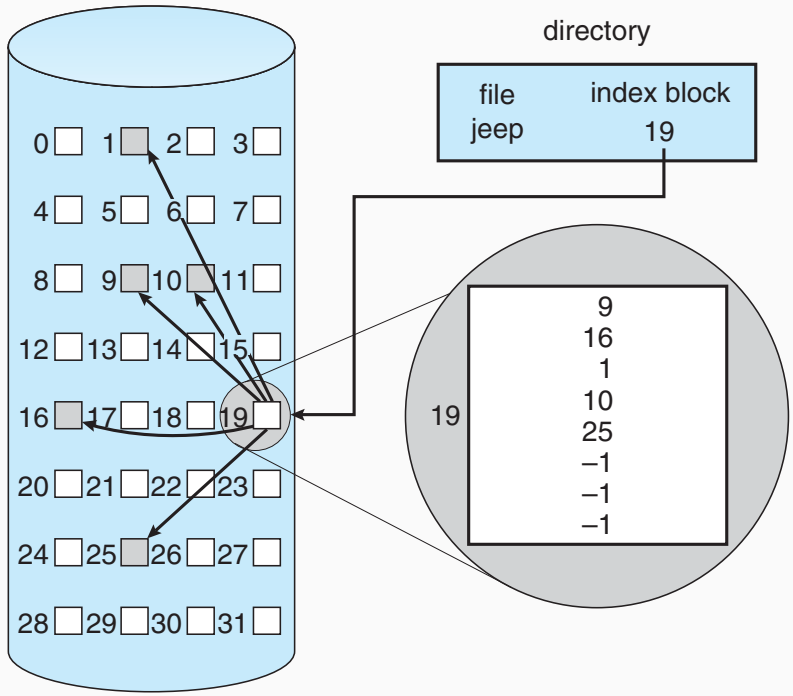
Asignación de bloques

- Archivos contienen *index block*
- *Index Block* contiene bloques del archivo
- Último cluster almacena EOF (*end-of-file*)



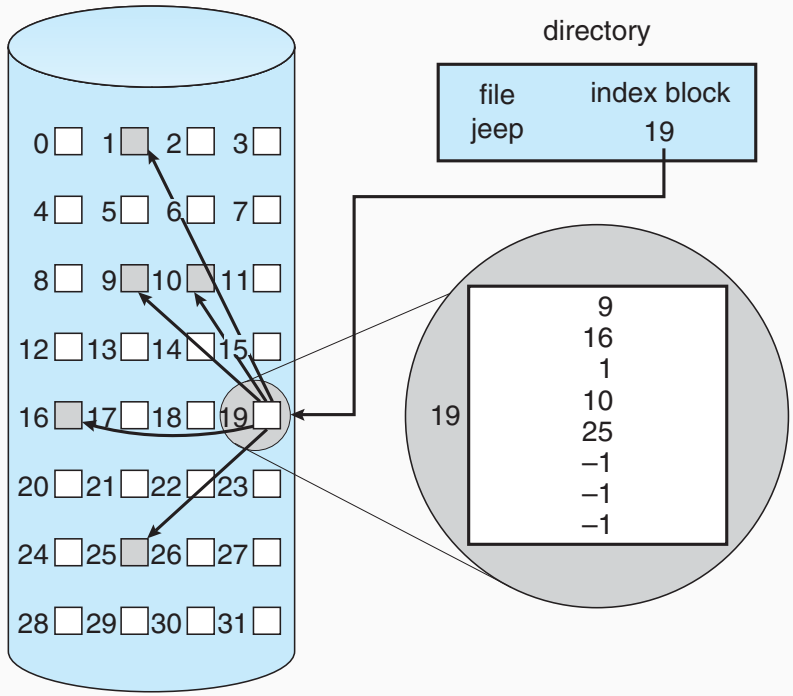
Asignación de bloques

- Archivos contienen *index block*
- *Index Block* contiene bloques del archivo
- Último cluster almacena EOF (*end-of-file*)
- +Acceso aleatorio sin frag. externa



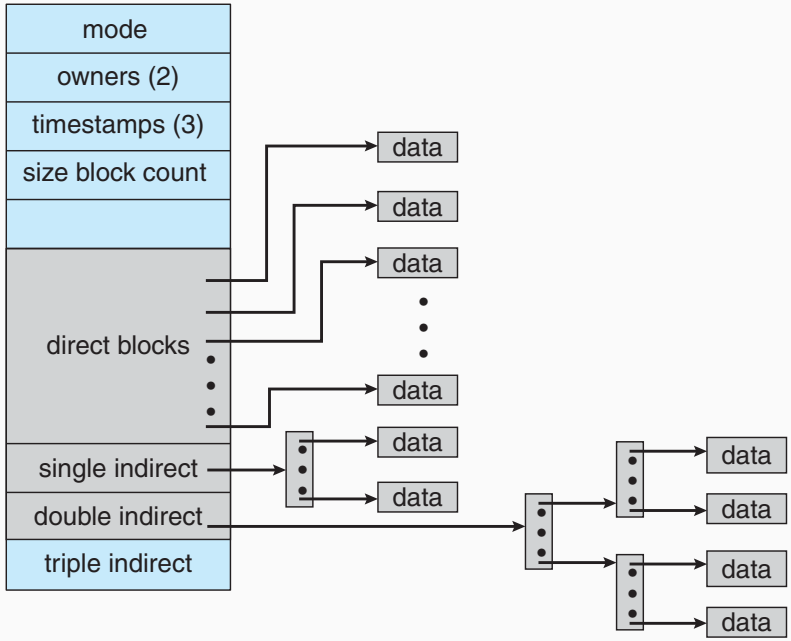
Asignación de bloques

- Archivos contienen *index block*
- *Index Block* contiene bloques del archivo
- Último cluster almacena EOF (*end-of-file*)
- +Acceso aleatorio sin frag. externa
- -Se pierde espacio por tamaño de *index block*



Asignación de bloques

- **Esquema enlazado**: última entrada de *index block* apunta a otro *index block*
- **Índice multinivel**: similar a tablas de página multinivel
  - Ej: blocks de 4KB, permiten 1024 punteros (de 32-bit)
  - Dos niveles permiten direccionar 1048576 bloques $\rightarrow 4$GB
- **Esquema combinado**: primeros p punteros son bloques directos, los siguientes apunta bloque de índice simple, los siguientes a



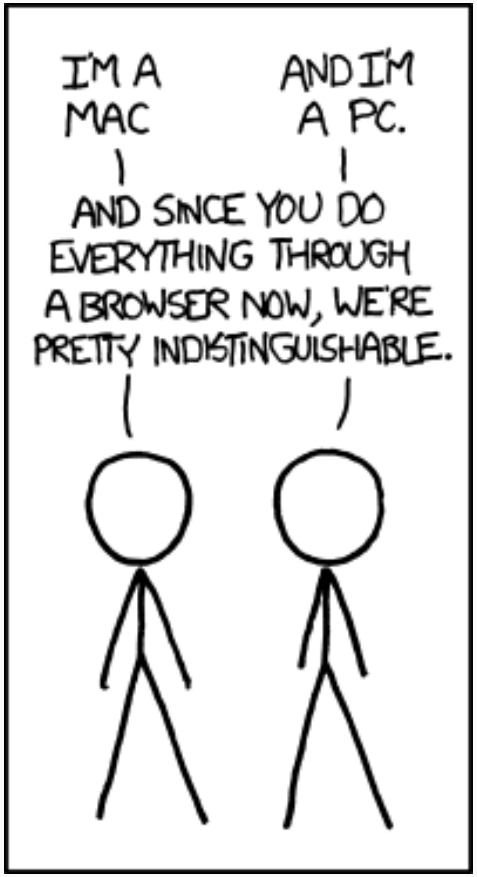
Finalmente...

Existe una multitud de sistemas operativos...



¿Cuál es mejor?

Finalmente...


I'M A MAC
AND I'M A PC.
AND SINCE YOU DO EVERYTHING THROUGH A BROWSER NOW, WE'RE PRETTY INDISTINGUISHABLE.