



→ Cesar Pedroza

capedrazab@unal.edu.co Ofi: 112

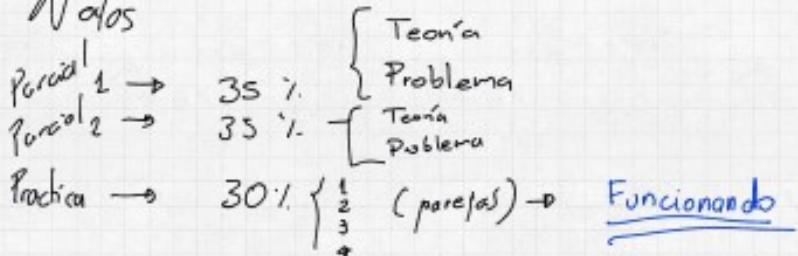
## Presupuestos

- C, C++.
- Arquitectura Computadores.
- Linux Console (Bash)
  - ↳ Acceso Remoto Para Configuración.

link curso

sites.google.com/a/unal.edu.co/sistemas-operativos

## Notas



S.O. Slackware

Silver Snark

Se puede entrar desde

# C y C++

## Tipos de Variables y Apuntadores

Estructura

< tipo de dato apuntado > \* < Identificador del apuntador >  
↓                            ↓  
int                           p

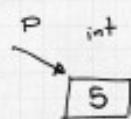
- Los apuntadores solo almacenan una dirección de memoria.

Inicialización de un apuntador

p = &i;  
↑  
operador para  
indicar la dirección  
de memoria

- Dos ó mas apuntadores quedan tener almacenada la misma dirección de memoria
- Para obtener el valor que esta almacenado hacia donde apunta un apuntador se usa el operador \*

$$\begin{array}{r} y = *p + 3; \\ \hline y = 8; \end{array}$$



Acceso de Atributos ó métodos con apuntadores

```
struct Data {
```

```
    char nombre[20];  
    int edad;
```

```
}
```

```
Data d;
```

```
Data *pd = &d;
```

```
(*pd).edad = 23;      }      mismo  
pd->edad = 23;      ]      diferente  
                        método.
```



2ebrero



Memoriayprocesador



7febrero



Directorio

Quiz de videos



## Voz\_5\_1

Taller 1

Memoria Dinamica

malloc()

se guarda de modo punteros.

# include < stdio.h>



Voz\_6\_1

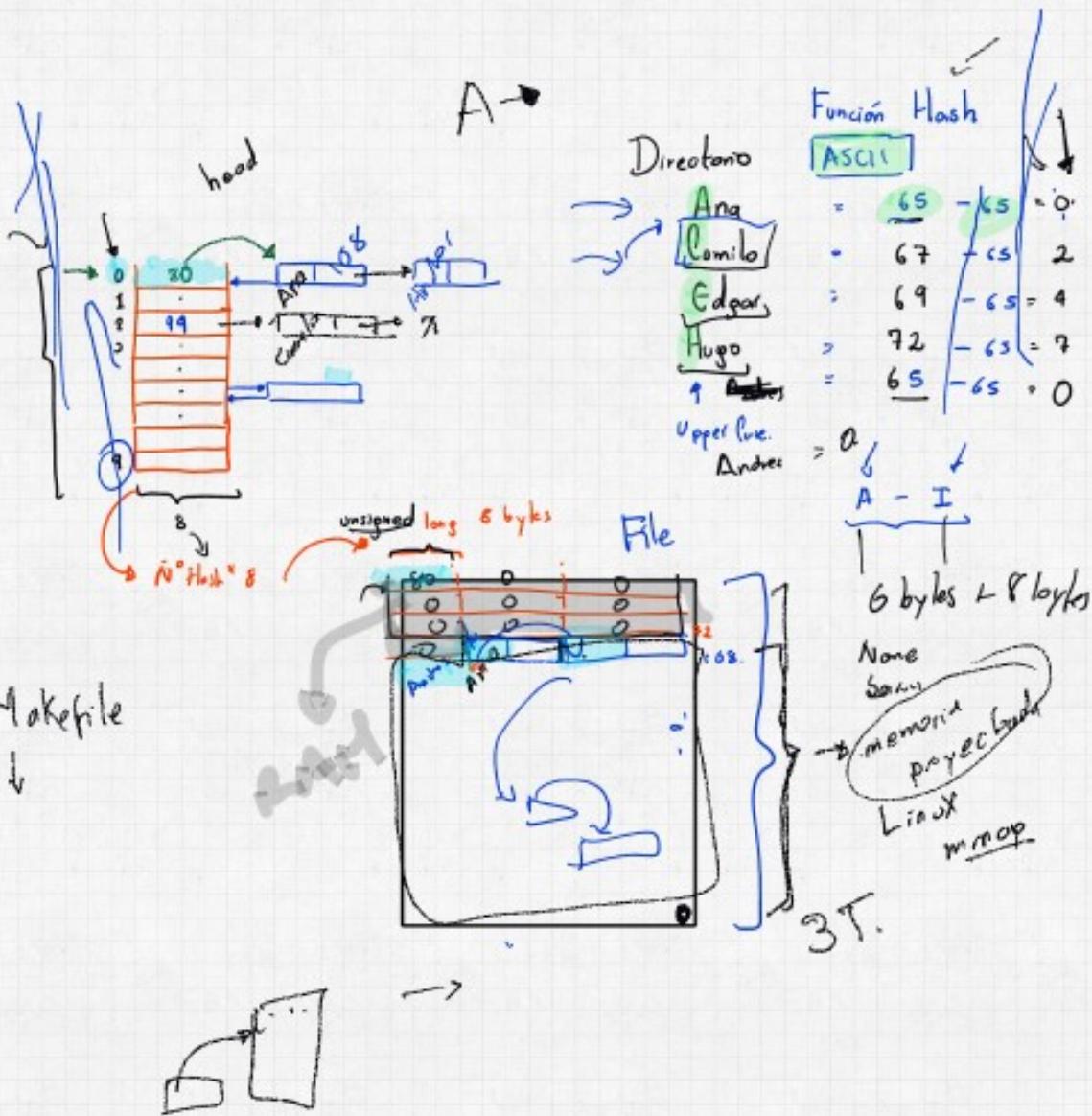
## Hash Table.

- ↳ Acceso Constante
- ↳ Almacenamiento lontino.

+

## Doble Linked List

- ↳ No acceso constante
- ↳ No almacenamiento lontino



## Hash Table.

- ↳ Acceso Constante
- ↳ Almacenamiento Continuo.

## + Doble Linked List

- ↳ No acceso Constante
- ↳ No almacenamiento Continuo

### Ejemplo:

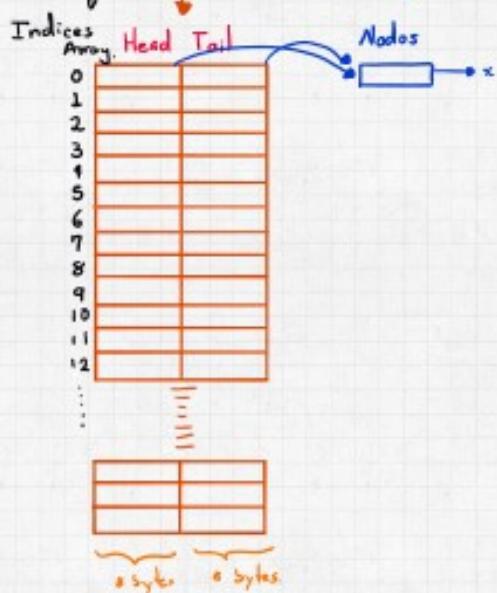
Directorio telefónico que necesita llenar rápidamente por letras sin importar mayúscula o minúscula. El directorio guarda 2 nombres de 10 letras y un teléfono de máximo 10 dígitos.

### Función Hash.

Usaremos la primera letra en mayúscula del nombre que vamos a guardar.

Primera letra con Uppercase.  $N^{\circ} \text{ ASCII} - 65 = \text{Indice "Array"}$

Graficamente.



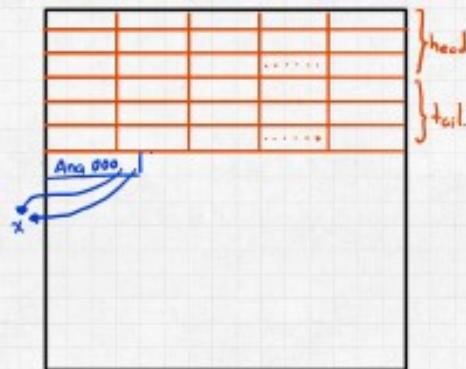
### El Nodo.

Contendrá un string de 10 letras, un teléfono que se almacenará en un entero + 2 apuntadores qd se almacenan en unsigned long.

Nombre teléfono apuntador apuntador apuntador

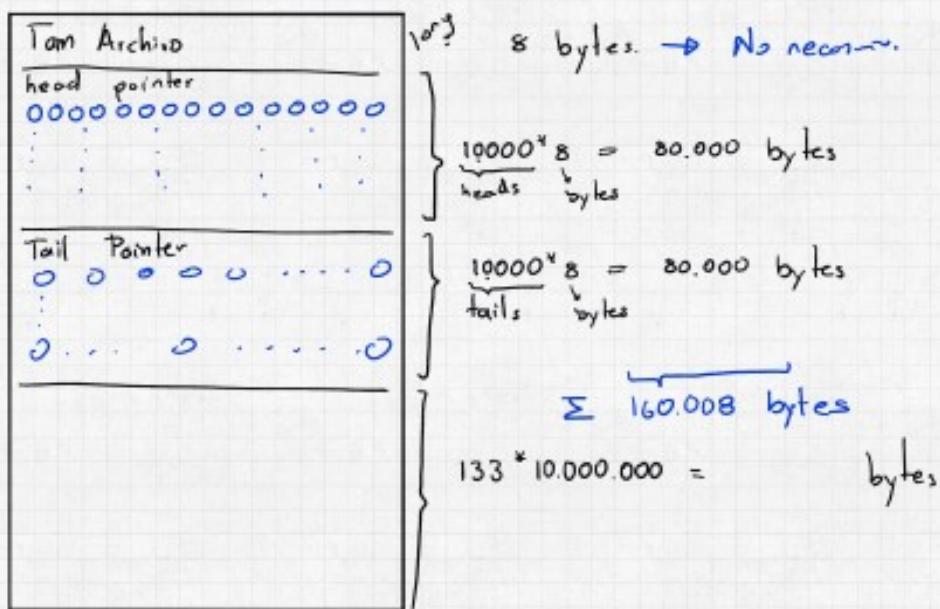
10 bytes 4 bytes 8 bytes 8 bytes = 30 bytes

Físico.



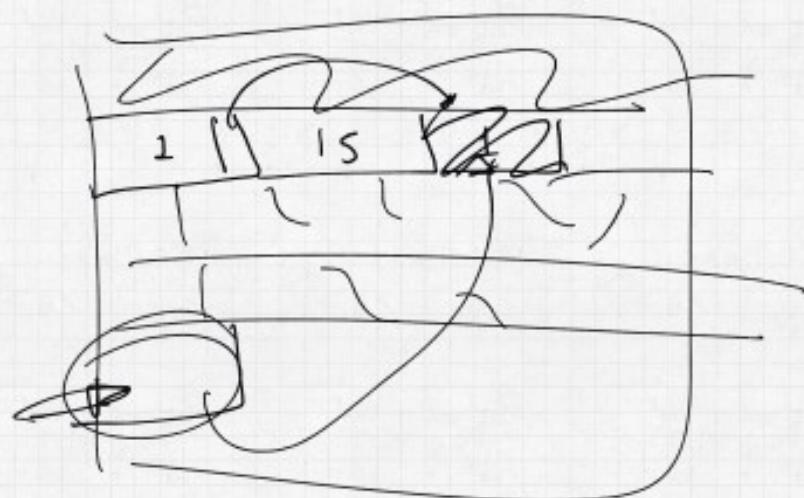
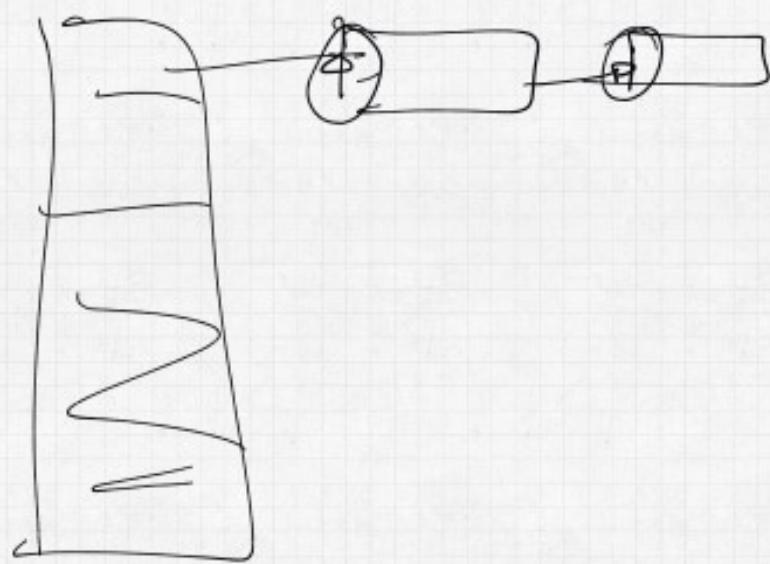
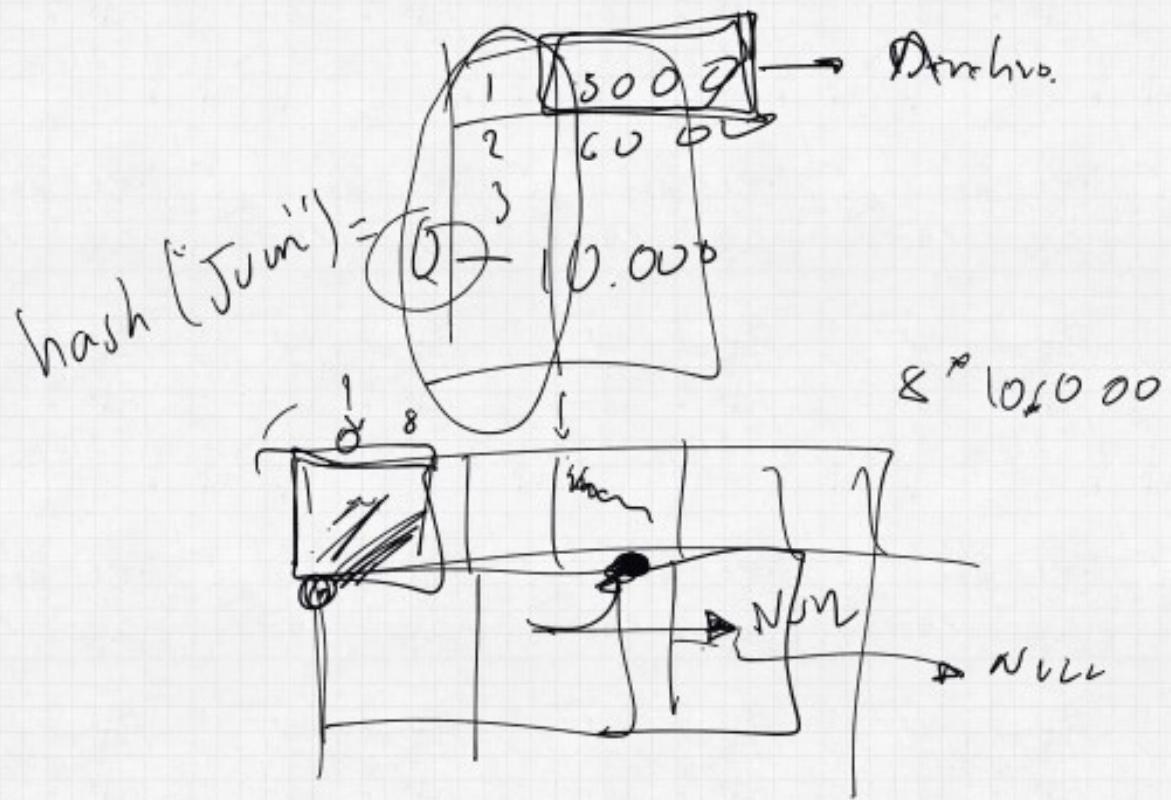
## Tutorial implementación de Hash table con dobleLinkedList

Init. data.dat

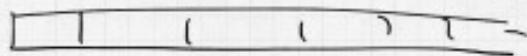
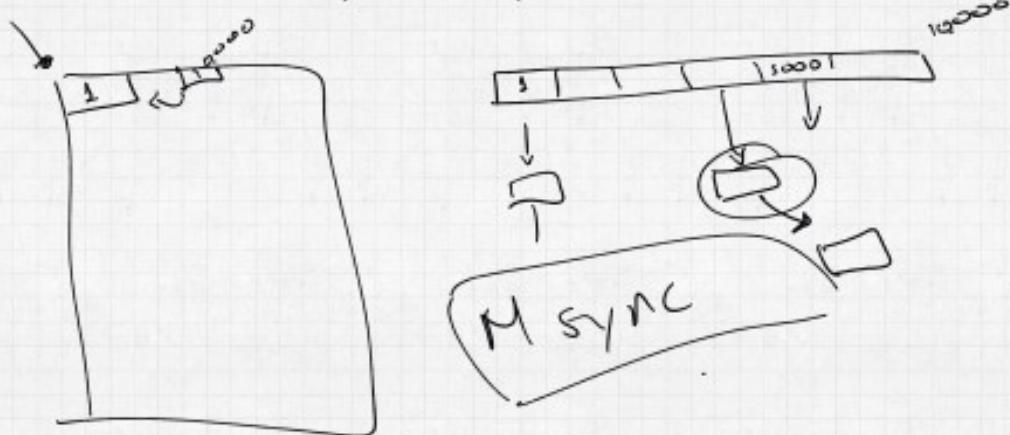


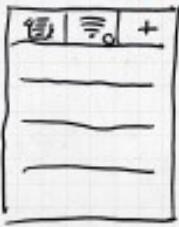
```
void *mmap(void *start, size_t length, int prot, int flags, int fd, size_t
```

↓  
bytes donde  
inicia la bucle  
del



```
caddr_t mmap ( void *start, size_t length, int prot, int flags,  
int fd, off_t offset);
```

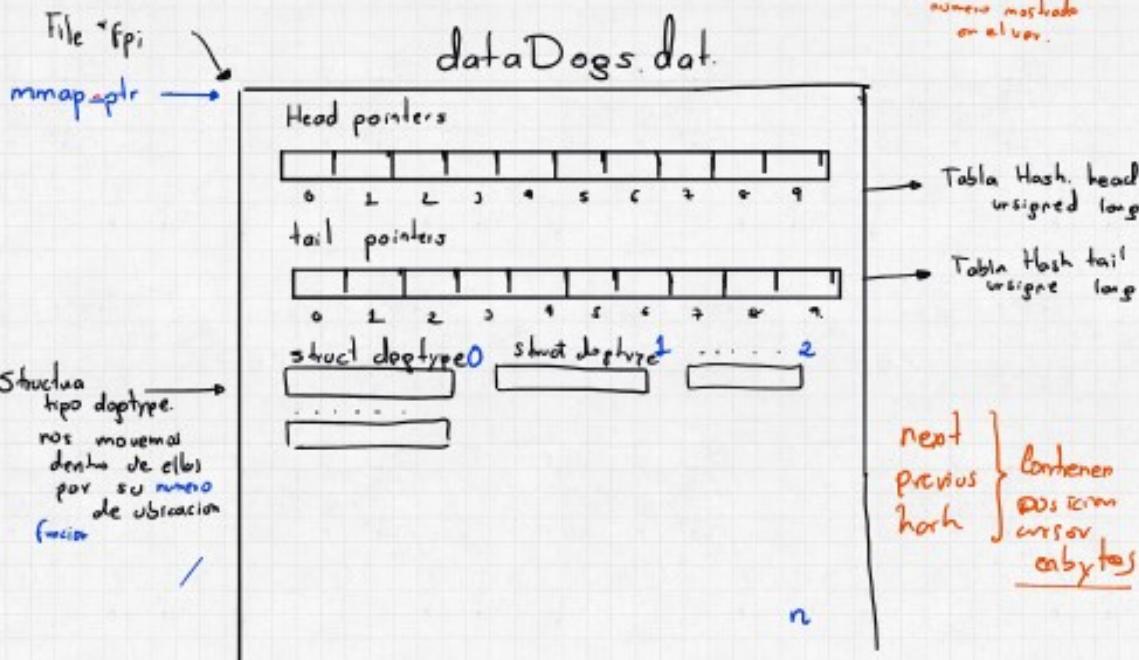




- 1. Agregar Teclado ↗
- 2. Editor de Teclado
  - 2.1. Teclado qwerty Inglés
  - 2.2. Otras teclas
- 3. Administrar Conexión

## Características de mmap.

- Ingresar Registro
  - Ver registro → Hash Hash  
Marker linked list
  - Borrar Registro → Quitar final item  
a posición del eliminado anular operador
  - Buscar Registro → ir con el numero mostrado en el user.



## Funciones:

Has dirige a la beta.

Me lleve con la cabecera  
directo a la cola.

Iteradora  $g'$  con el nombre y usando el hash muestra y encuentre cierto nombre.

función para moverme en el archivo send ()  
 send(-2) → Inicio fin ()  
 send (int) → posición dentro del archivo  
 send (-1) → inicio archivo

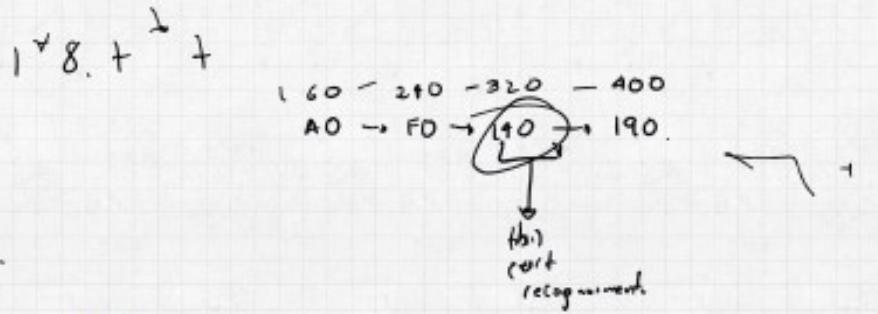
### sucherbar Hash.

Le day directions

```

Reynera in Dogan
{
    char nombre[32],
    char Tipo[32],
    int edad,
    char raza[16],
    int estatura,
    double peso,
    char sexo
}
unsigned long next
unsigned long prev

```



Borrar perro

deleteDog ( direcciónPoco )

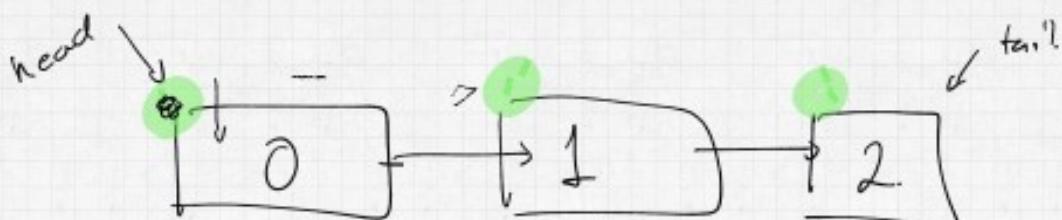
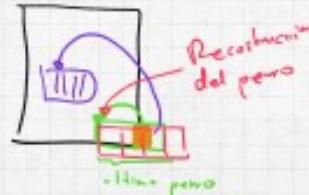
send(2)

capturamos ultimo perro

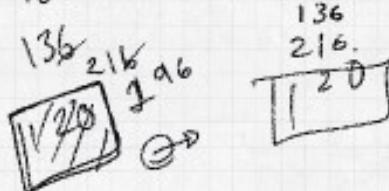
reconstrucción poco.

reestimulando el perro anterior.

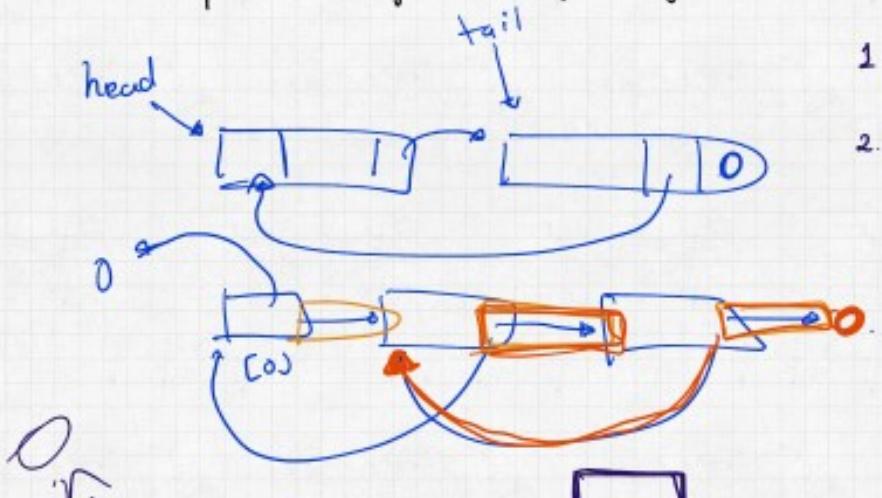
actualizandolo el Perro → next va a ser el g  
vamos a eliminar.



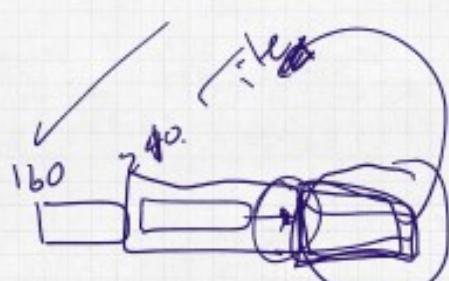
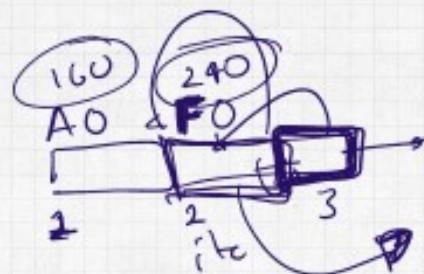
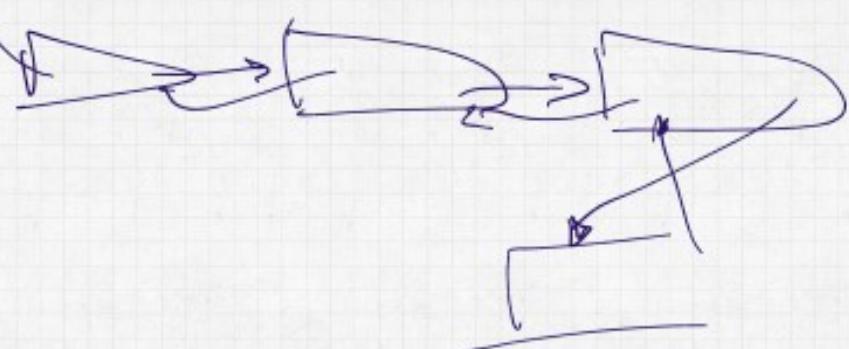
160 - 200 - 320



Function  
updateHash ( long Rank, unsigned long adr)

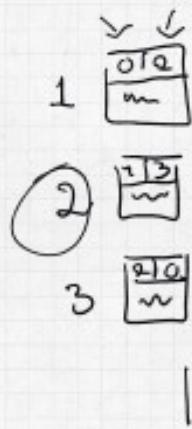


1 160 ✓  
2 240, 1  
64  
1 G  
3 0  
160 ✓  
240



FO





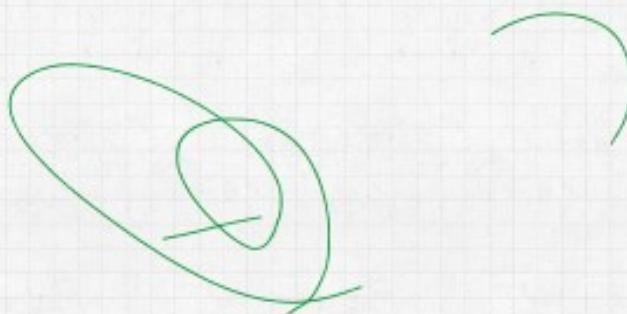
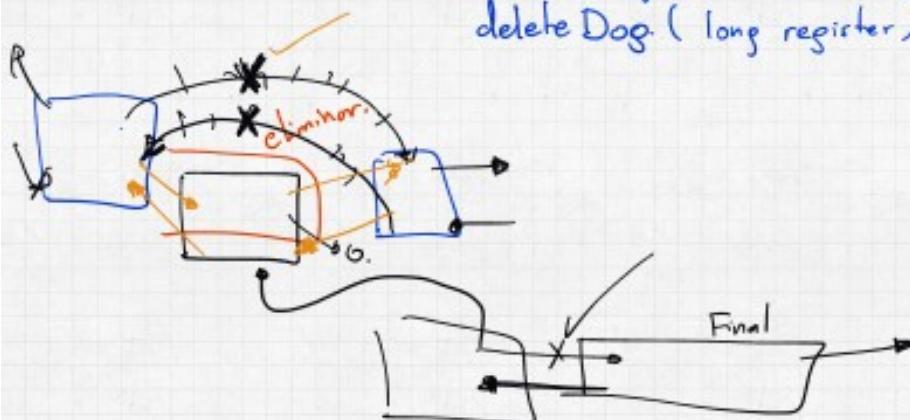
~1



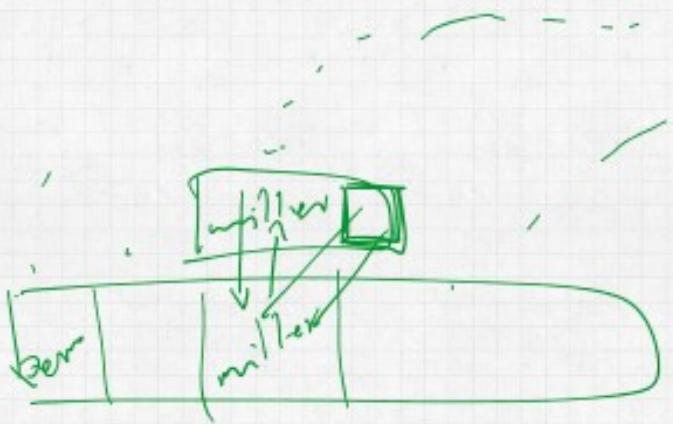
0

y  
G

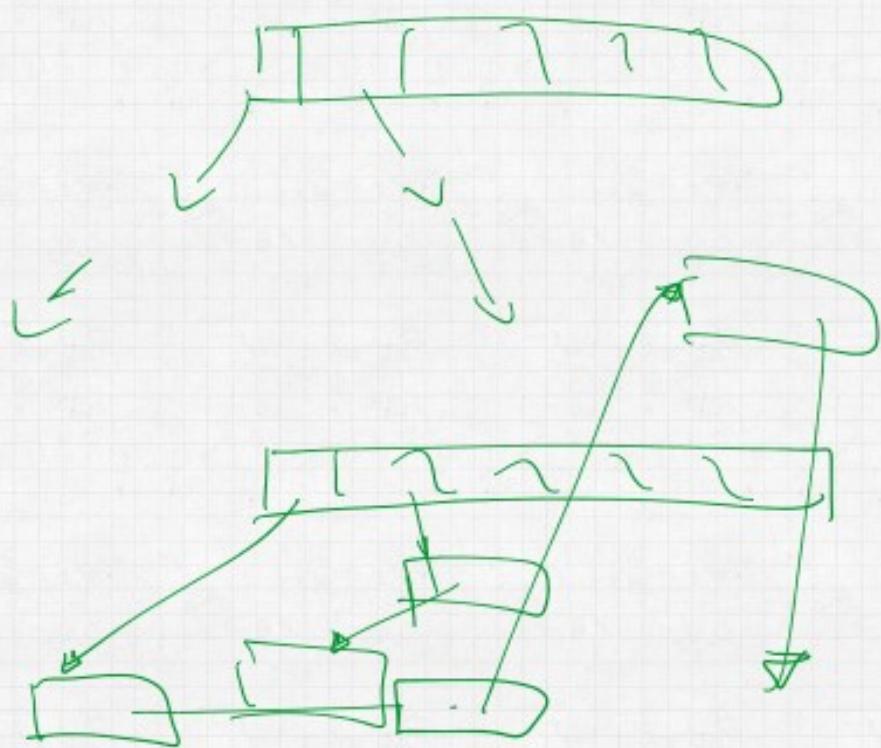
Borrar Registro  
delete Dog( long register){



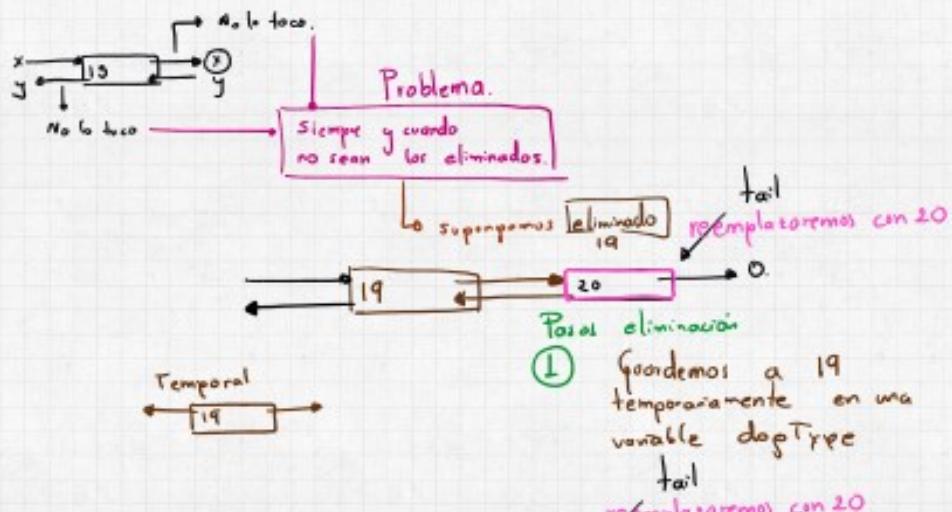
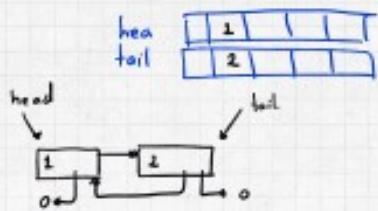
Ridley & 9373



Codig 4 (M111e)  $\Rightarrow$  3  
 Peso  $\Rightarrow$  1  
 Nuller  $\rightarrow$  3

## Analisis eliminación

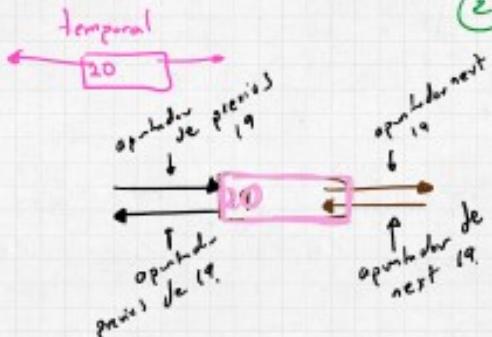


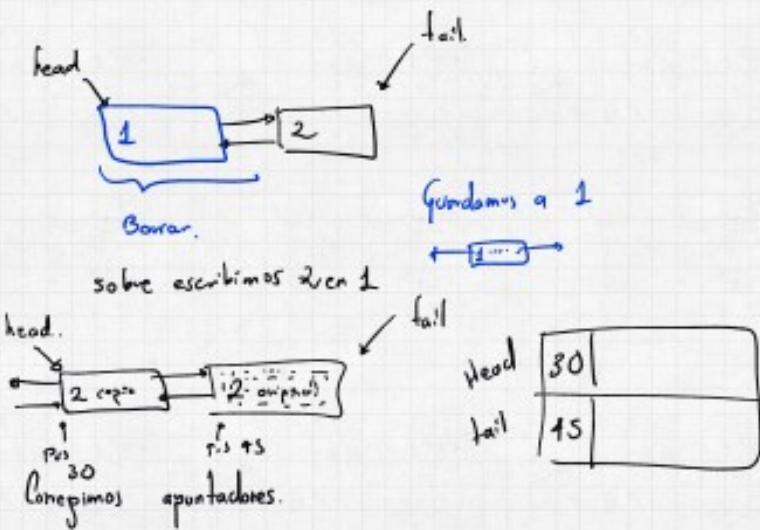
① Guardemos a 19 temporalmente en una variable `dogTree`

~~tail reemplazaremos con 20~~

② Guardemos a 20 temporalmente ???

escibamos a 20 en el lugar de 19.



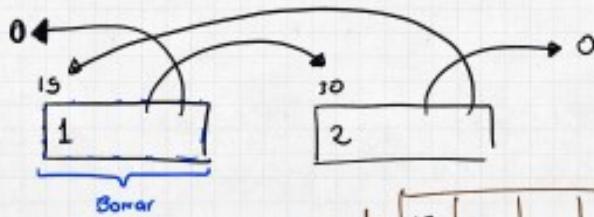


if (*i*.previous == 0) { es cabecera

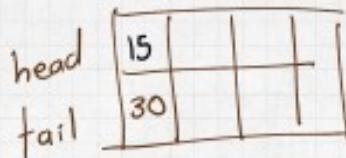
actualizamos la hash  
de esta linked list con la  
nueva posición calculada  
de donde se encuentra

45.	
.	

Dirección  
Punto



$f(y, y)$

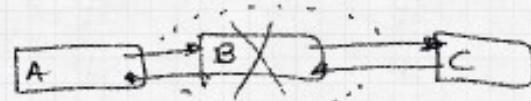


Guardamos 1  $\boxed{1 \ 0 \ 30}$   
previos next

Cogemos el previous de 2 y lo reemplazamos por el de 1.

Copiamos 2 en 1.  
 $\boxed{2 \ 0 \ 0}$   
previous next

Verificamos si 2 es una cabeza o cola y actualizamos



$a \xrightarrow{x} 2$  Jelte



## My hash Function

"a" length 1 → size of  $\Rightarrow 1$ .  
"az" length 2 → size of (char) = 2.

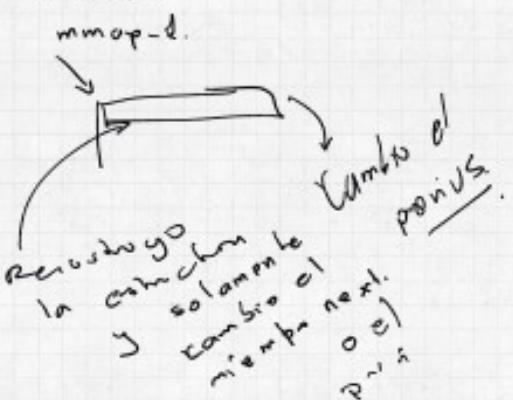
```
int const multiplier = 263
long const prime = 1000000007
long hash (char name[32])
{
    long hash = 0;
    int i;
    for (i = 31; i >= 0; i--) {
        hash = (hash * multiplier + name[i]) % prime;
    }
    return (hash / SIZE_HASH) * 8;
```

Ingresar perro

struct dogtype inDog( File \*fp, unsigned long position )

- ↳ -1 agregar nuevo perro
- ↳ mayor -1 reacomodar perro en la posición contenida.

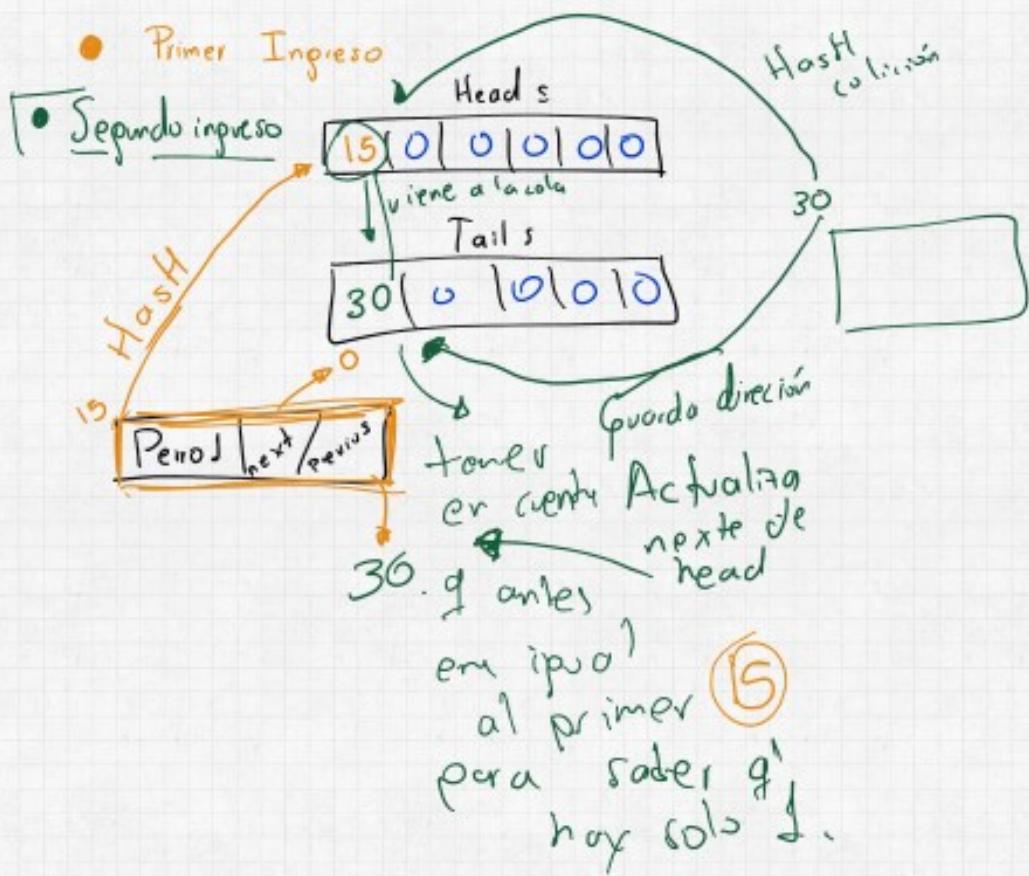
Cambio de .next estructura ya existente.



changeNext (Nuevo Next, Nuevo  
Previous)

## Hash y Linked-list

12





Voz\_8\_1



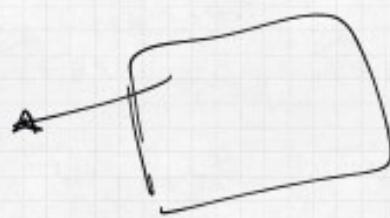
Tra.1

13

15

15

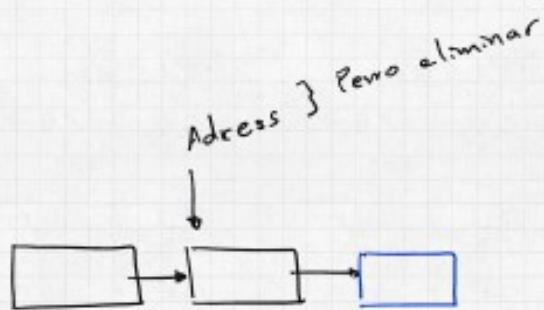
Frente int.15



A

address  
↓  
[delet]

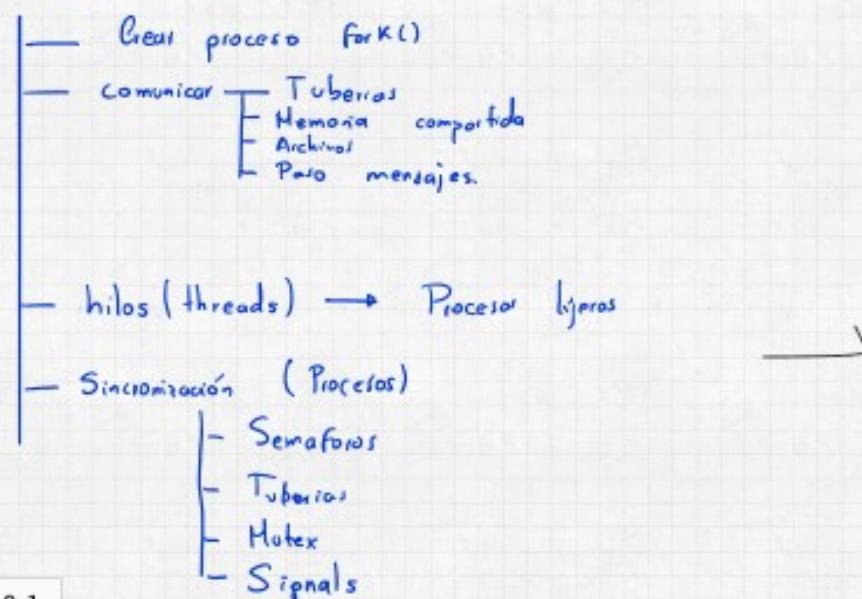
dogAp  
[ ]



Final address  
↓  
[moe\_dog.]

## II Intro

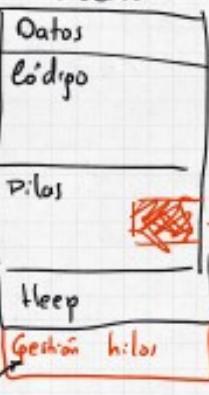
### 1. Procesos



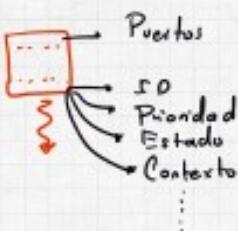
Voz\_9\_1

## Hilos

### Proceso



### hilos



## hilos

### POSIX

pthread.h

pthread\_create()  
pthread\_join()

```
pthread_t hilo;
```

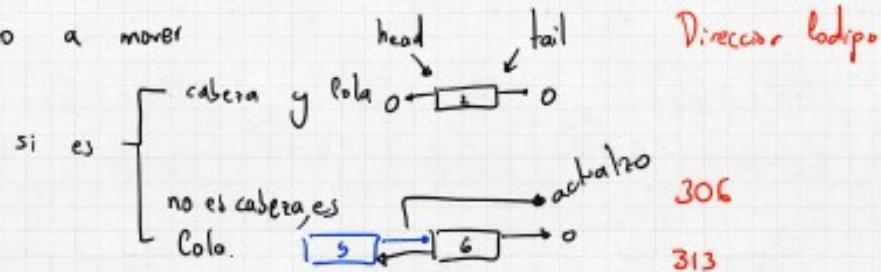
```
pthread_create(&hilo, NULL
```

Compile and Link?  
with -pthread

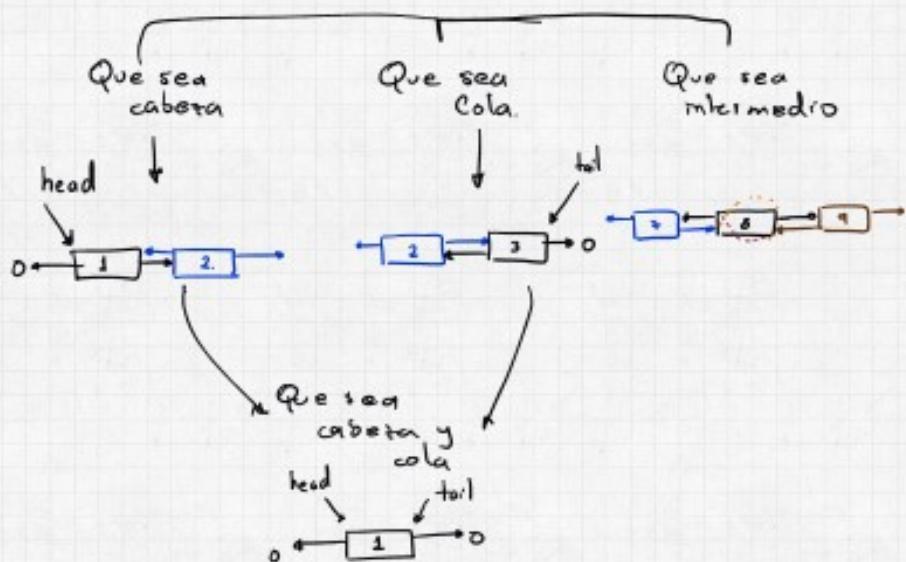
## Possibles ocasiones.

- Q el perno a mover sea dirigido  
el eliminar sea único
- 

### - Acciones Perno a mover



### - Acciones perno a eliminar.



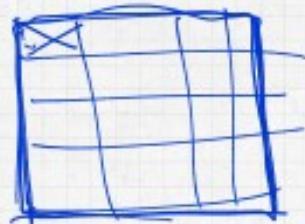
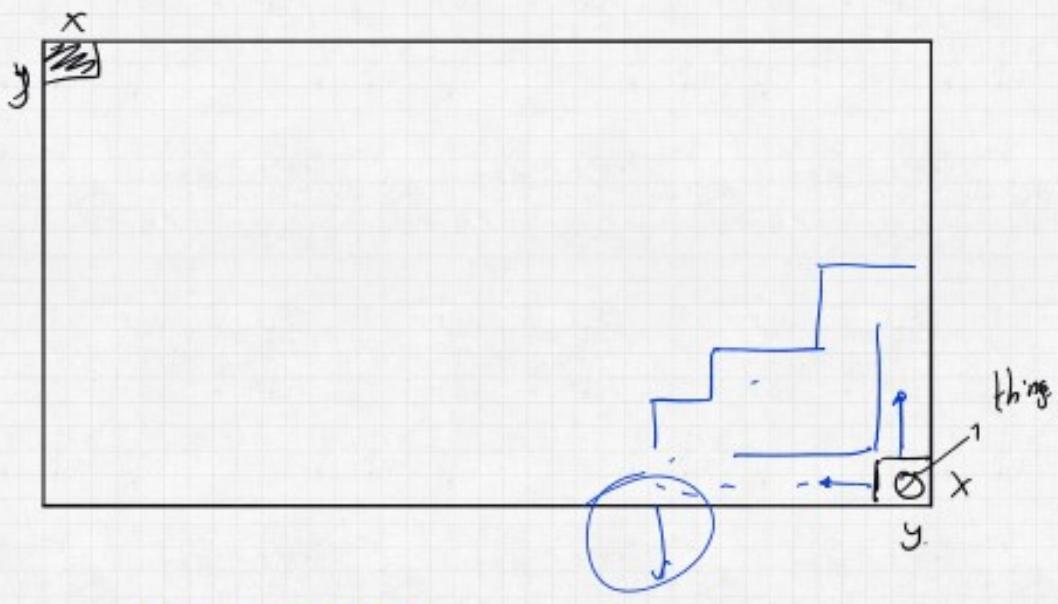
Número.

— — — — —

Dígitos

— 0 — 0 — 0 — 0 — 0 —

6 1 3 2 1



for  $c_i = o_j \quad i <$

## Tarea

Revisor semáforos POSIX

- semaphore.h
- sem\_open()
- sem\_close()
- sem\_wait()
- sem\_post()

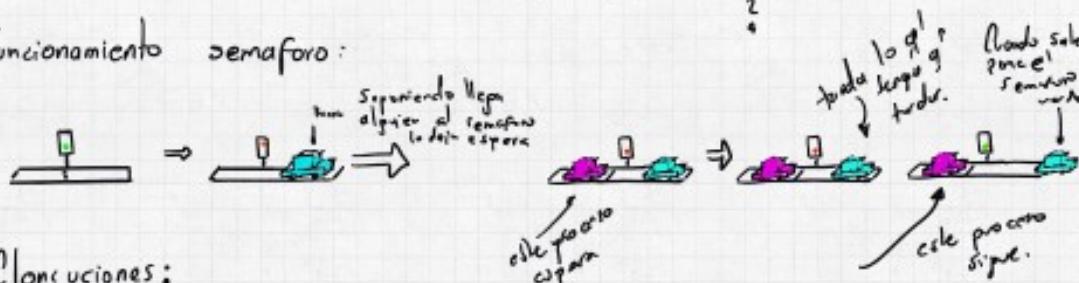
man :

Bajar código: <http://pastebin.com/4mN4UKBN>

Páginas MAX PROCESOS

1  
2  
4

Funcionamiento semáforo:



Conclusiones:

Para 1 proceso  $\Rightarrow$

El scheduler no saca al proceso hasta q' este se termine de ejecutar, aunque este tarda mucho. y el semáforo está en rojo para el segundo proceso.

Para 2 procesos  $\Rightarrow$

El scheduler saca al primer hilo del procesador debido a q' este se demora mas de lo normal. luego entra el otro proceso e imprime se repite la operación cada vez q' este se tarde.

## Procesos

- intro
- creación (fork...)
- comunicación →
  - Archivos
  - Tuberos
  - Mem comp
  - Pole mensajes.
- hilos

- sincronización
  - Semáforos
  - Tuberos
  - Mutex
  - Signal.

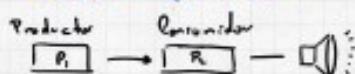
Sincronización

Voice\_11\_1

v Semáforos.

Problemas sync Procesos Problemamemocom

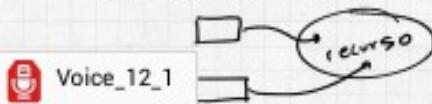
Productor - Consumidor → Solución con semáforos.



→ Sem-wait();  
→ uso recurso  
→ sem-post();

Lectores - escritores

- \* Varios escriben a un recurso



Es mejor dejar  
que solo uno lo haga.

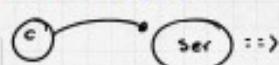
Método sync

→ Tuberos

→

- Client - Servidor

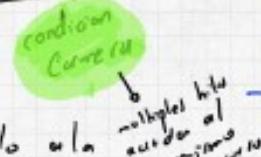
Voice\_12\_2



Este es el reporte de secuencia para que los dos no se quedan en un recurso

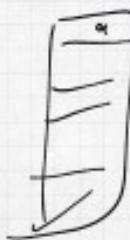
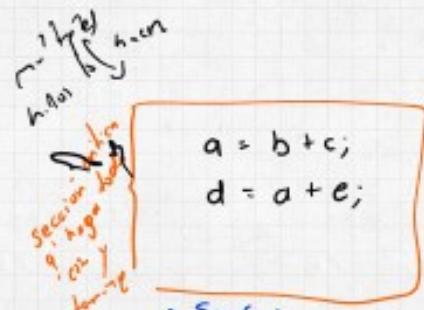
- Secuon - libro

1 segmento de código  
sea ejecutado por 1 hilo ala vez

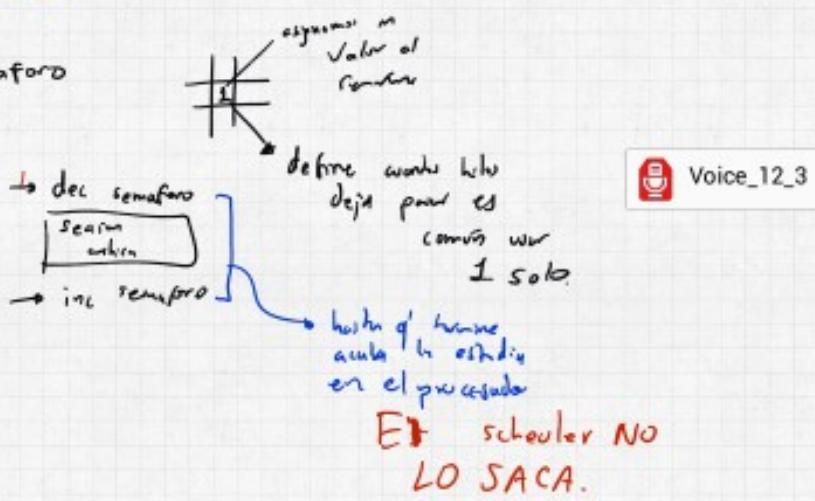


Para cada opción

→ Problema de Programación Multihilo



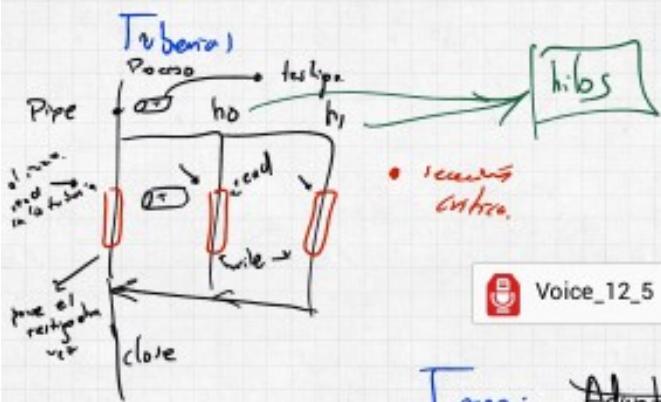
## Semaforo



MAX PROCESOS → ~~que~~ en el  
 codijo.

Voice\_12\_4

→ Phil 3.



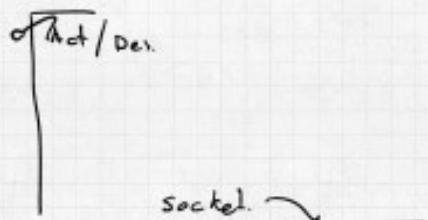
Voice\_12\_5

Tarea: Adaptar el

Sección Práctica con el

- ✓ Tuberías + mutex + semáforos.
- ✓ Taller piping + MUTEX (1)

Taller [ mutex  
Signal ]



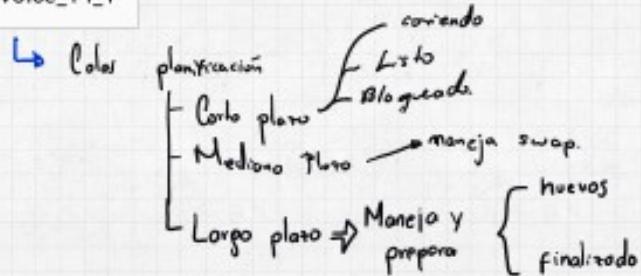
ctrl + c  
Signal administrada por el S.O.

signal();  
Configurar la detección de un  
señal creando

setsockopt = Reuso de puertos.  
del socket

Scheduling. → Planificación { Posibilidades multitarea.

Voice\_14\_1



Voice\_14\_2

→ criterios planificación

fFlush()  
reiniciar  
repetir

previsibilidad →  
REAL TIME OPERATIVE SYSTEM

modo de decisión

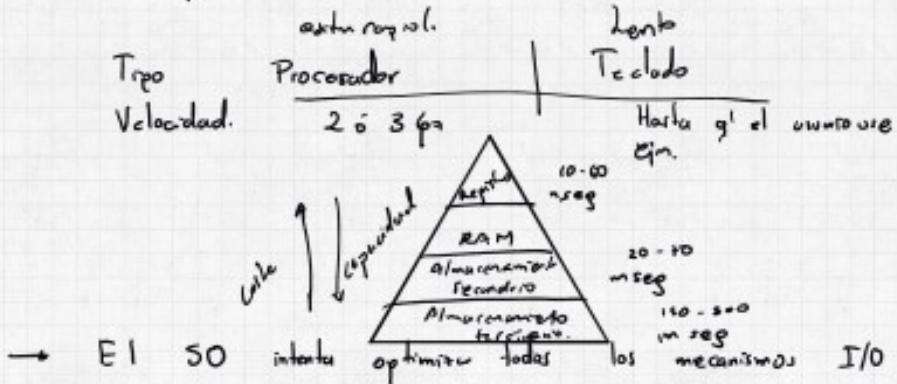
no apropiarse  
apropiaciones.

FIFO Come First Served



## Sistema de archivos

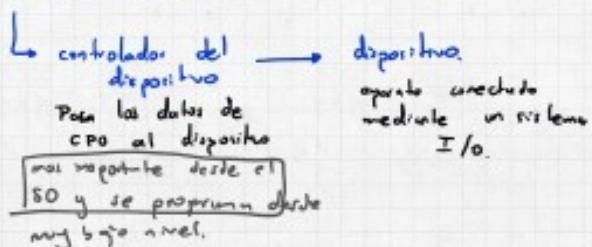
- Los dispositivos I/O son un cuello de botella en la computación actual.



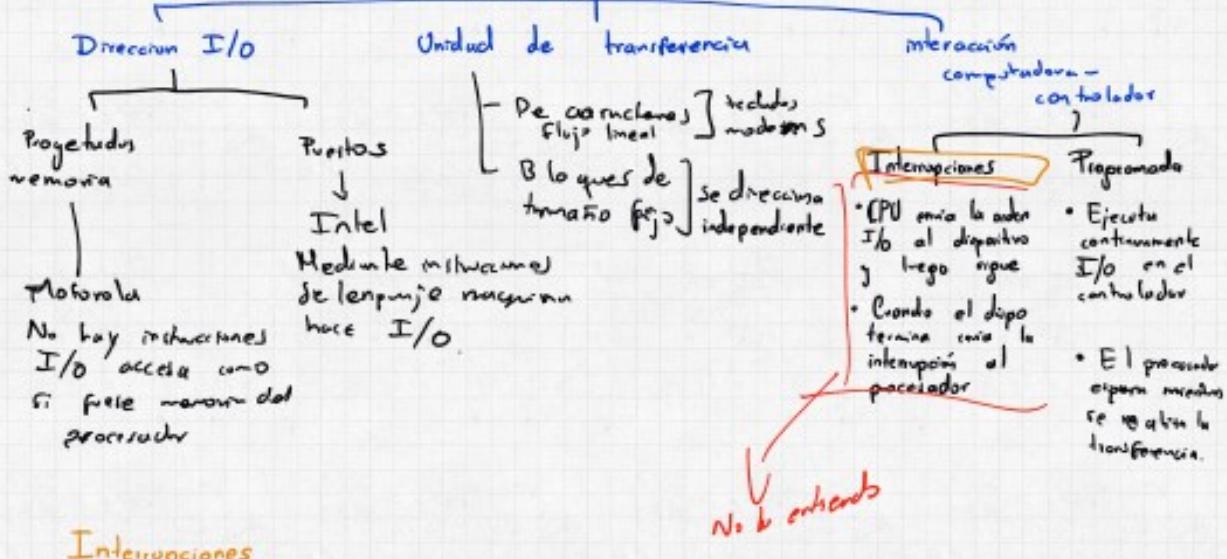
## Sistema de entrada salida.

Busca facilitar el manejo de estos mediante una lógica y simplificada de estos.

- Conexión de un I/O.



### Atención del hardware del dispositivo



### Interrupciones

- Modelo ligado a la arquitectura del procesador
- Usualmente medianas interrupciones
  - vectorizadas
  - enumerables
- Si existen varias señales de solicitud de interrupciones code una tiene su prioridad
- La rutina de atención maneja las interrupciones cuando se producen.

Atributos Archivo unicos

Tipo de archivo y Protocolo.

Propietario

Grupo del propietario

Tamaño archivo

Fecha modificación

Tiempo de eliminación

Número de vínculos (soft / Hard)

Unix Nombre inequívoco 4096 caracteres

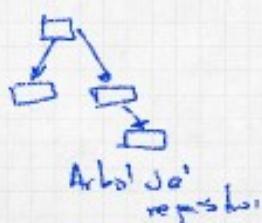
Estructura lógica de un archivo



Byte o registro  
de longitud  
fija.



registros  
de longitud  
variable



Arbol de  
registros

Directorio

Cabeza  
Todos objetos  
Cubierta  
Modulos  
objetos

Unix el número mágico identifica a un archivo como ejecutable.

Mínimas partes de un ejecutable

- Cabeza
- Texto
- Datos
- Información de cargo
- Tabla de simbólos

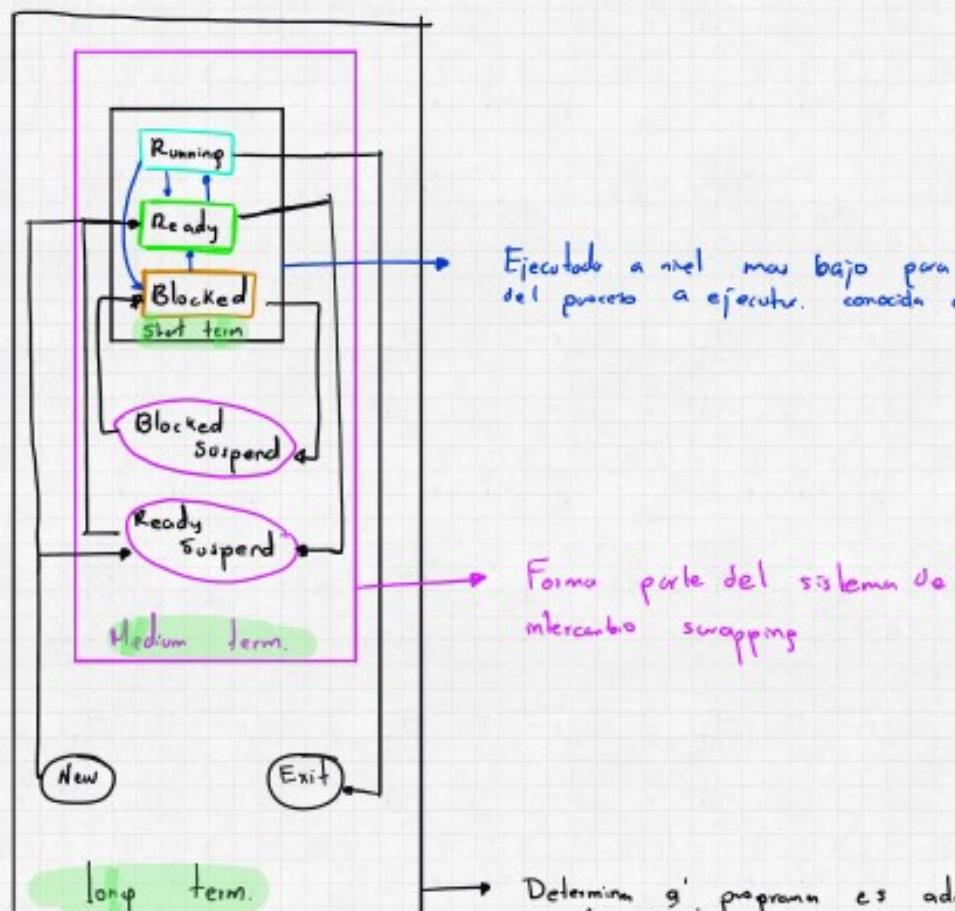
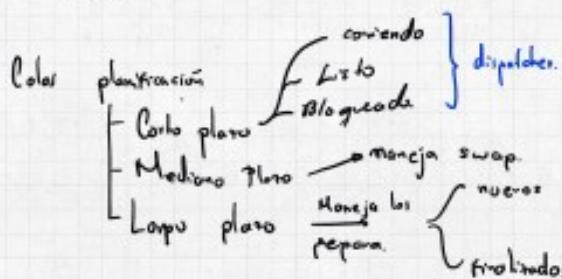
Tamaño de Bloque óptimo es de 8k para un ancho de banda óptimo  
y una de disco.

Enlaces duros

ln \_ Ficherodelenlace \_ Nombreenlace

## The beautifull scheduling

- \* Estrategia del SO para compartir la CPU entre procesos.  
requisitos a cumplir
  - tiempo de respuesta
  - procesamiento
  - eficiencia
  - otros.

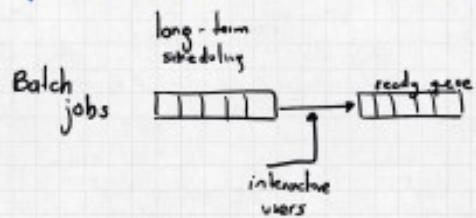


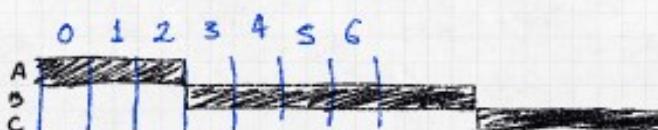
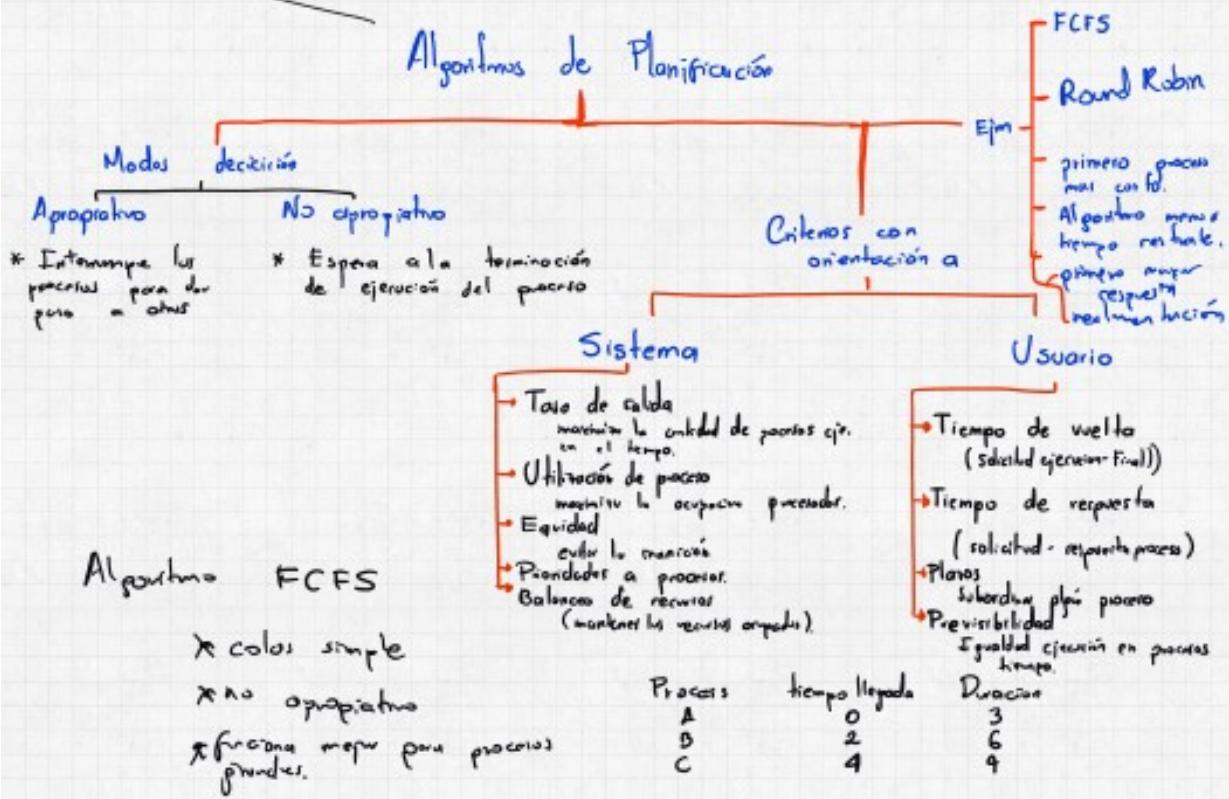
Ejecutado a nivel más bajo para la selección del proceso a ejecutar, conocida como dispatcher

Forma parte del sistema de intercambio swapping

Determina si el programa es admitido para ejecutar, y trae información en proceso. Puede ser aceptado o rechazado.

## Diagrama de colas en la planificación

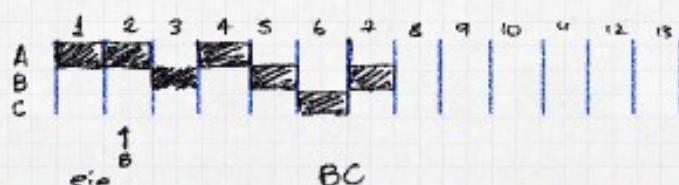




Algoritmo. Round Robin o Fracciones de Tiempo

\* Apropiație, menținând el rechap

UN PROGRAMA  
PUDE SIMULAR  
ESTO ✓



Usar la **cola** simple para no equivocarse tener en cuenta el tiempo de **procesamiento**

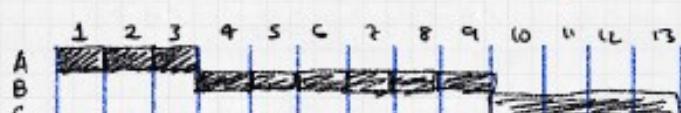
Algoritmo "primero el proceso mas corto."

x Se selecciona el proceso con menor tiempo de ejecución.

x No operativa

x difícil predicción del tiempo ejecución.

→ mas usado en real time  
operable system  
puesto q̄l cosa +  
tiene de  
acceso



Proceso	Tiempo llegada	Duración
A	0	3
B	2	6
C	4	4

Algoritmo "menor tiempo restante"

- x Solo es la versión asignativa del algoritmo más corto.



Algoritmo "primero el de mayor tasa de respuesta"

- x prioriza el proceso q' mas espera el CPU

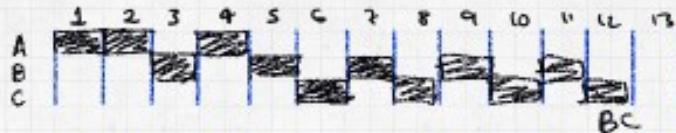
$$R = \frac{w + s}{s}$$

↑                      |  
 tiempo de espera    tiempo de servicio  
 R = w + s         s  
 ↓                      ↓  
 esperado            ejecución

Algoritmo realimentación

- x Degrada la prioridad de un proceso al pasar por una cola de nueva
- x Los procesos cortos se ejecutan mas rápido

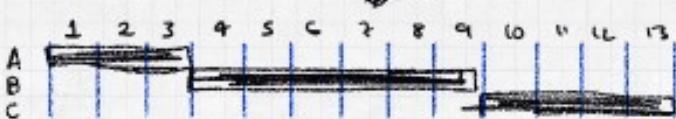
Proceso	Tiempo llegada	Duración
A	0	3
B	2	6
C	4	4



round robin  
 $p=1$



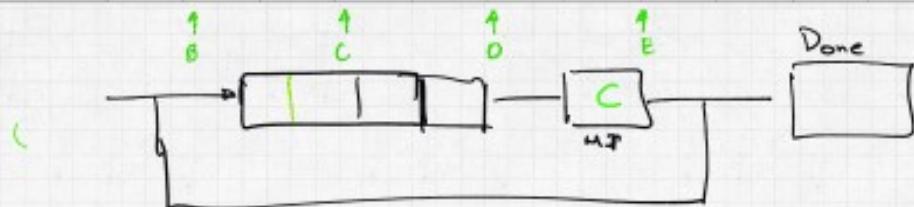
Primer ms corto



FCFS

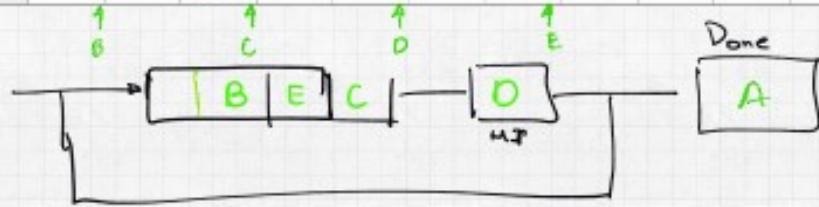
algoritmo  
mínimo  
tiempo  
restante

	1	2	3	4	5	6	7	8	9	10	11	2
A	m	m	m	m	m	m	m	m	m	m	m	m
B		m	m	m	m	m	m	m	m	m	m	m
C												
D												
E												

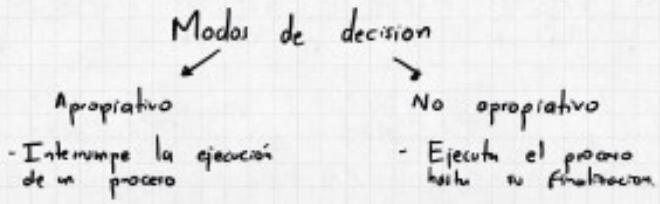


D

	1	2	3	4	5	6	7	8	9	10	11	2
A												
B			m									
C				m								
D					m							
E						m						



D



El Kernel de Linux posee el algoritmo más justo  
Completely Fair Scheduler.

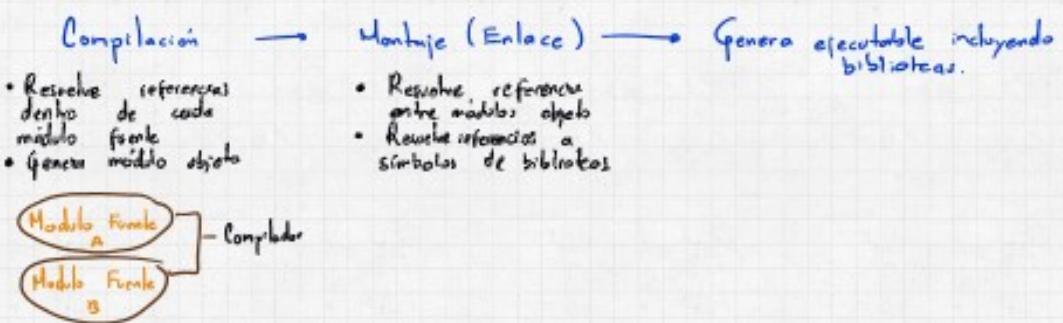
Brian Fucking Scheduler.

Recompilate Kernel in Virtual Box

↓

Para probar el  
poner y quitar  
modulos

## Fases en Generación de Ejecutable



bool → 0  
 bool → 0  
 Anita Guadagna  
 for (int i = 0; i < 3; i++)



while = for = do while

for ( ; ; ) {

4 bytes.

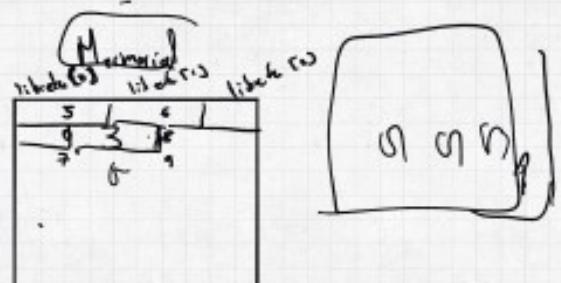
char \*a;  
 int border;  
 double double;  
 char + - int;

while (1)

00000000.

1000  
 0101

2,148,483,643  
 9 byte → int  
 ↓  
 32  
 01 214

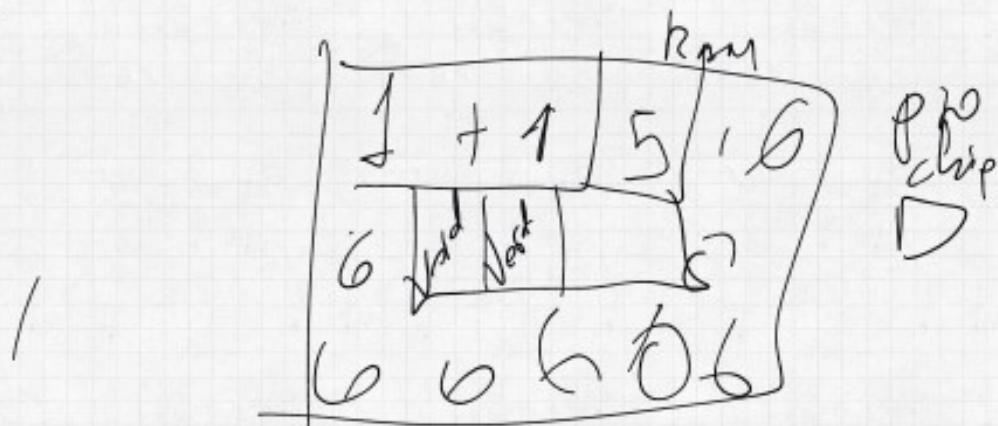
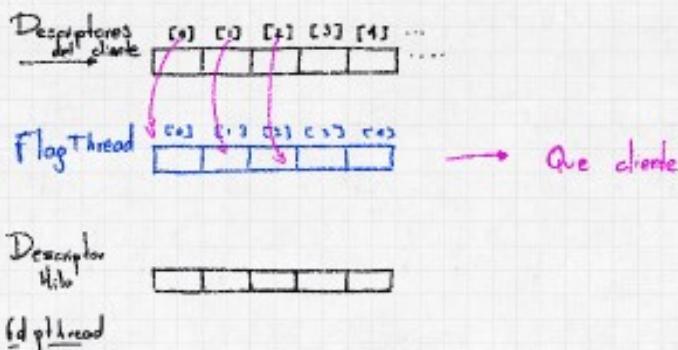


$$\begin{array}{r} 999 \\ \times 448 \\ \hline 999 \end{array}$$

Aneglo se receptionan **id** de los clientes.  
 ↓  
**client fd.**  
 Aneglo  
 6  
 6 fd 0 1  
 todos los hilos  
 (0,0000,0000,0000,0000,0000,0000,0000)

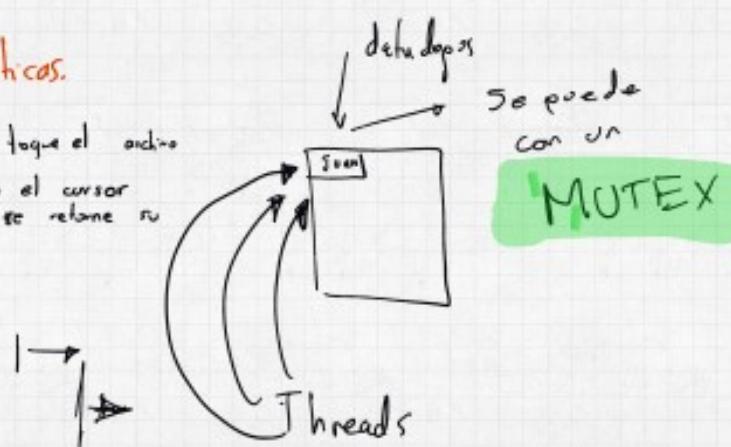
Aneglo de Control Hilos      **flag Thread** → Puede simbolizar 1-- hilos vivos.

Manejo mediante 2 Aneglos

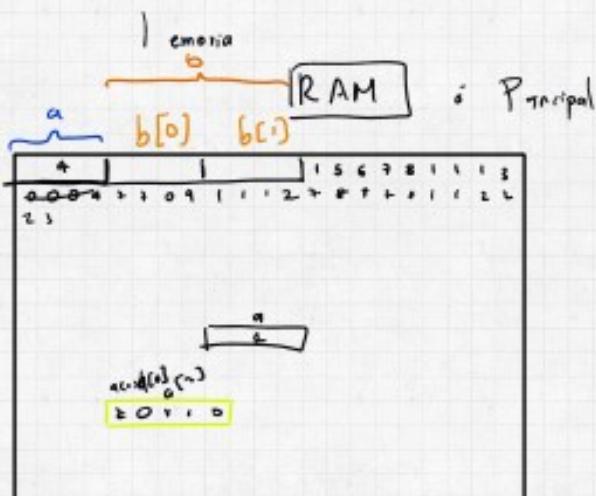


### Secciones Críticas.

- Cada vez q' un hilo toque el archivo
- Cada vez q' se mueva el cursor dentro del archivo y se retorne su ubicación
- 
- 



`pthread_mutex_init( pthread_mutex_t *`



int a = 4;  
int b[2];

b

$$a + f =$$

$a \neq b$  char = String  
if ( ) { }  
 $\downarrow$

$$a + 1 \quad 1 + 1 = 2$$

$$a + 1$$

97 → 97.00

5 x 3  
3 1  
3 1

char z[5];  
 $\downarrow$   
 $a[0] = z[0]$ ;  $\Rightarrow$   
 $a[1] = z[1]$   
 $a[2] = z[2]$

$$1 \times 10^{-1} =$$

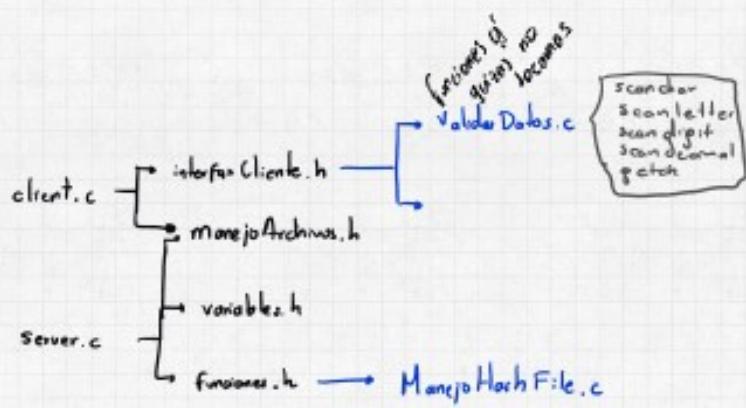
$$\frac{1}{10}$$

$$1^2 = \frac{1}{1}$$

$$1 \times 1^1$$

$$\frac{3,9}{10^{38} \times 1}$$

## Distribución del programa



# Práctica 2

Server gestiona  
serverDogs.log

$$1 \text{ byte} = 8 \text{ bit}$$

1 log = Fecha YYYY MM DD T HH MM SS  
5 char 32 bit 32 bit 32 bit 1 char 32 bit 32 bit 32 bit 32 bit  
5 byte 4 byte 4 byte 4 byte 1 byte + byte 9 byte + byte = 30 byte

Archivo recibido por el server.

Código de la operación char	Sig mensaje a recibir
inserción 1	Struct Dogtype
lectura 2	Número de rep a ver
borrado 3	Número de registro a borrar
búsqueda 4	String a buscar (32 bytes)
Salir 5	→ <i>mueve el hijo q' atendio la solicitud.</i>

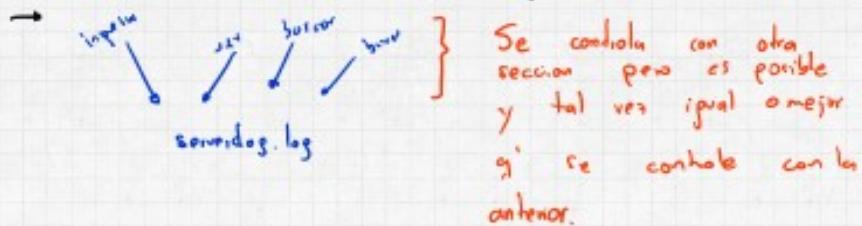
Secciones críticas ↗

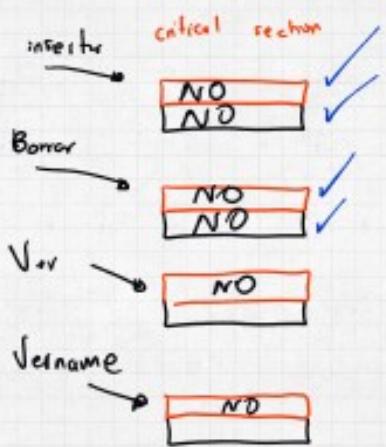
→ DataDogs.dat

- al insertar mueve hash y archivo (aprendo)
- al borrar mueve hash y archivo (aprendo)
- al ver usa la hash y navega entre posos
- al buscar con una cadena recorre desde la cabeza hash la colia toda la linkedList mediante (hash)

→ Serverdog.log

→ Para cualquier número de msg log



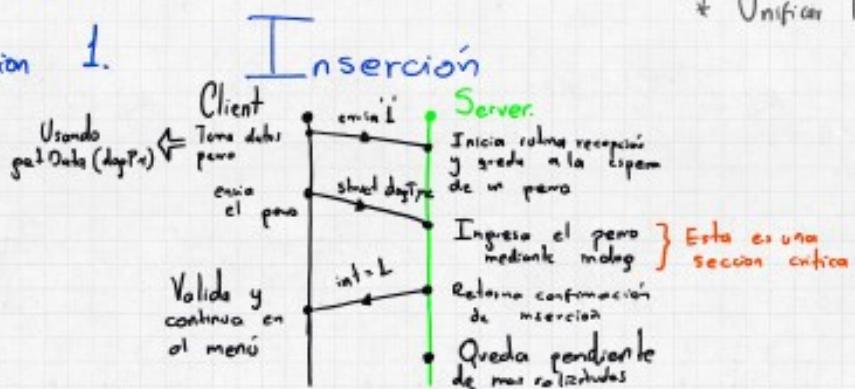


# Diagrama de Bloques

Falta

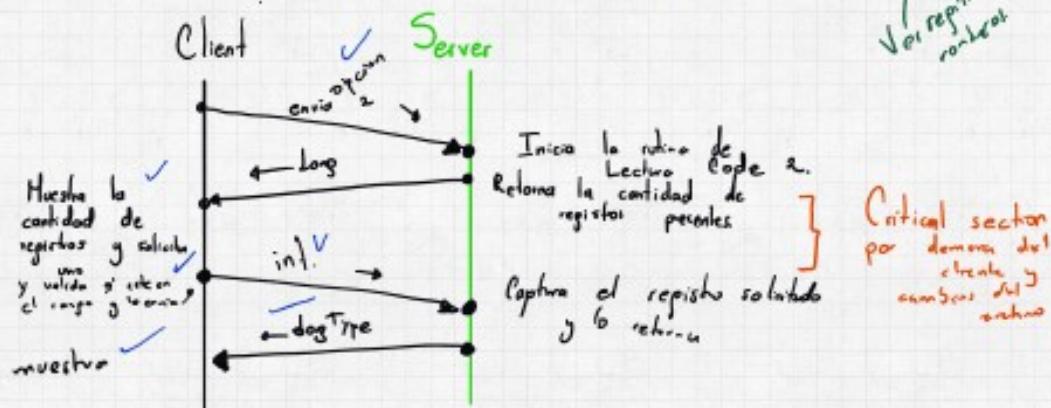
\* Unificar la

Opción 1.

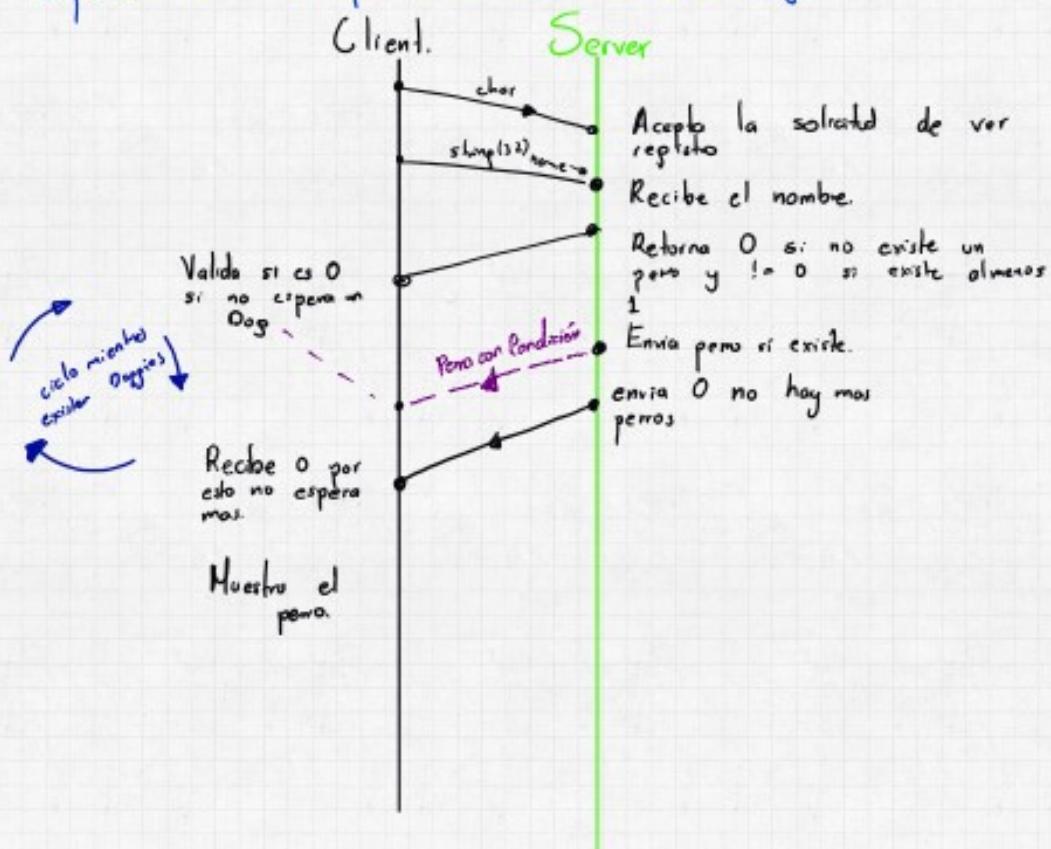


lock  
unlock  
View des  
Ver registro / control

Lectura 2 "mediante el número de Registro"

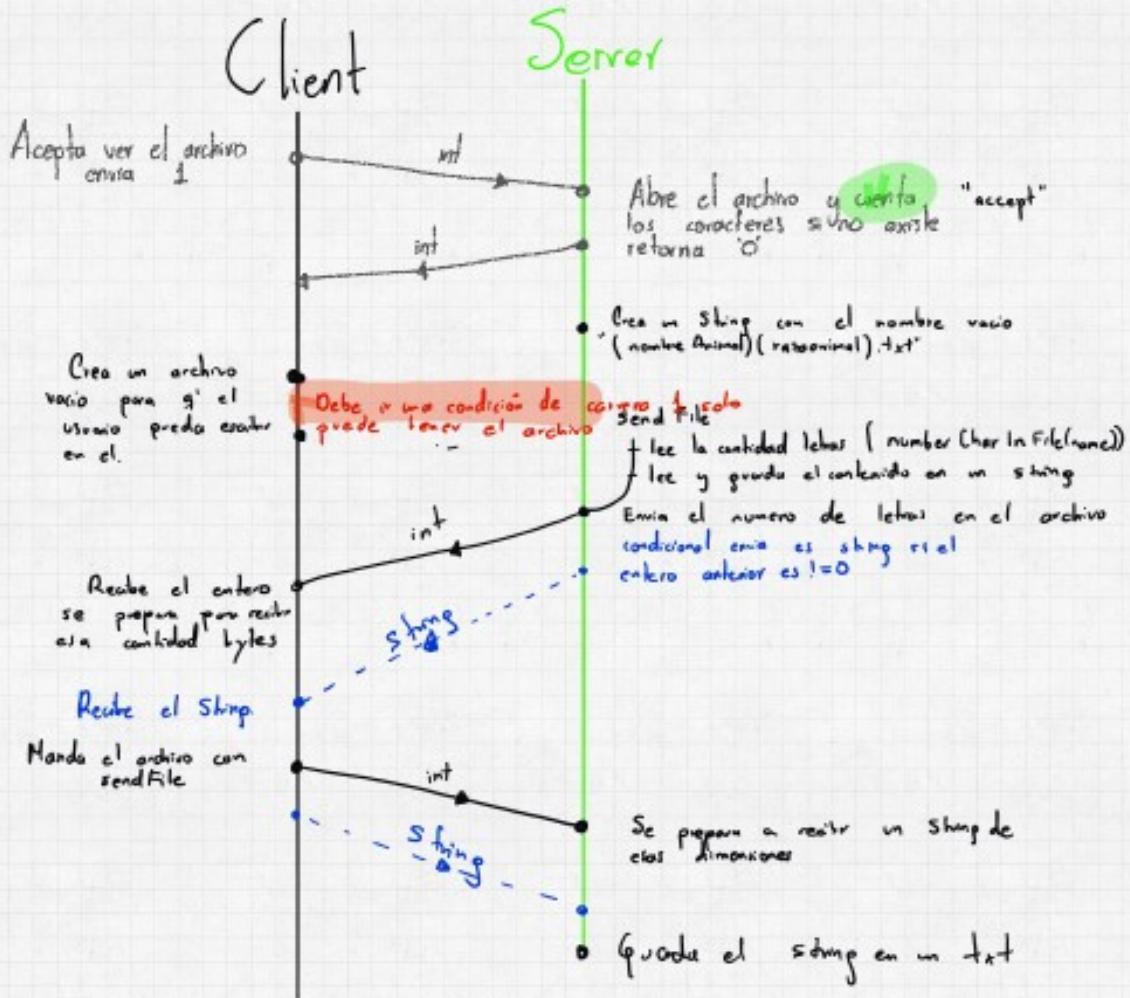


Opción 3 "Ver perro con un nombre de reg"

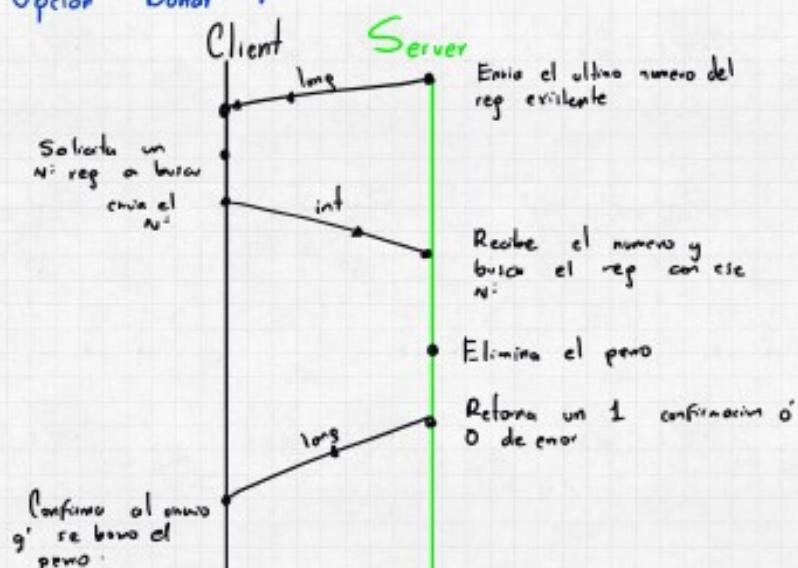


## Funciones para el envío del Archivo.

readFile. → Retorna un string con los caracteres dentro del archivo.

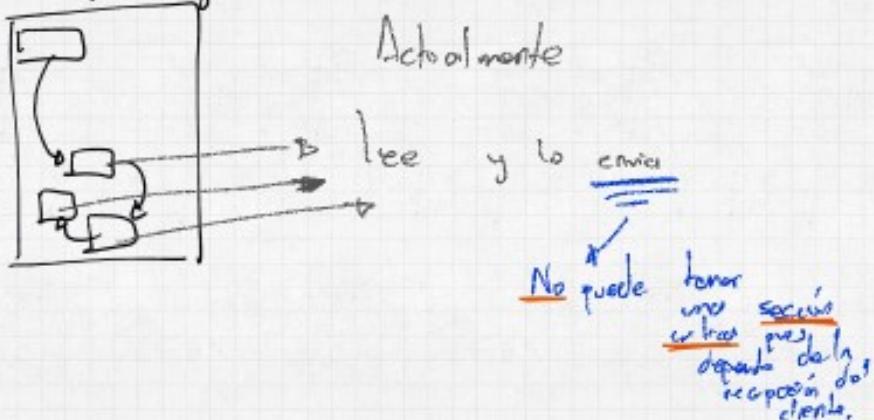


## Opción Borrar 4

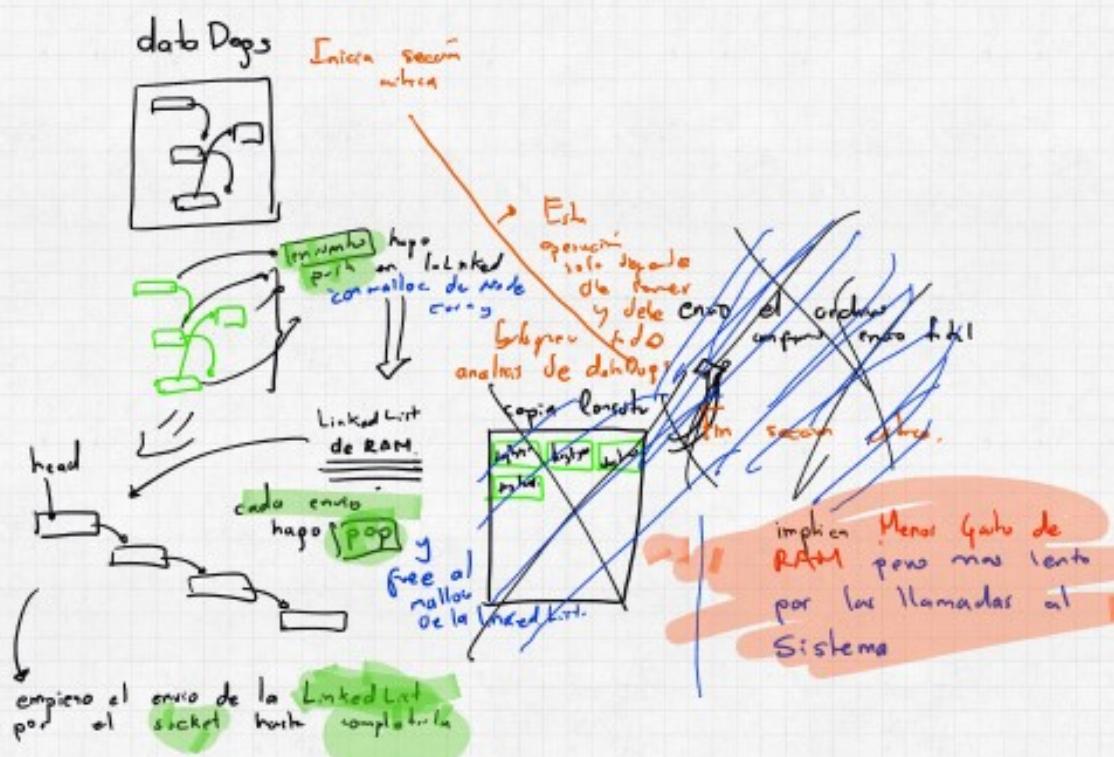


Sección Crítica al ver una cantidad de registros.

dataDogs.



Objetivo.



Funciones solución envío de un archivo.

→ enviar con el descriptor archivo y el descriptor del cliente.

recv → espera hasta recibir Parámetros totales de lo esperado.

-1 error

0 existen un cliente  
descendido.

Punto

Nº → importando byte por  
13 pu

12 → 2, 4       $\frac{1}{4} < 0$

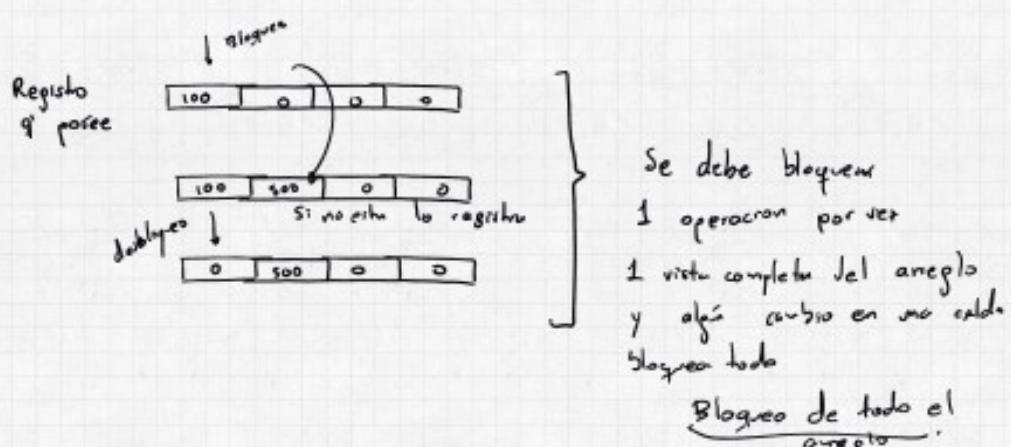
No me  
divide.

establecio max 4.

$\frac{12}{4} = 3 \rightarrow$  tengo q realizar  
3 servidas conectar.

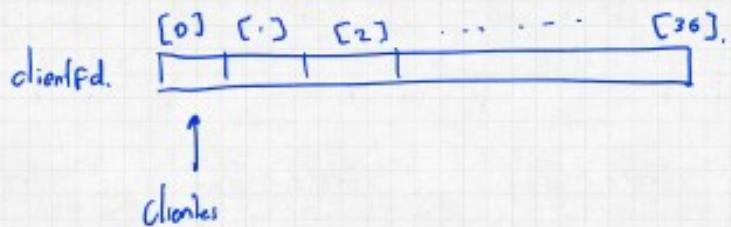
→ recibir y dejar en el descriptor de un archivo y con el  
descripción del cliente.

Envío de 1 solo archivo. → parámetros.



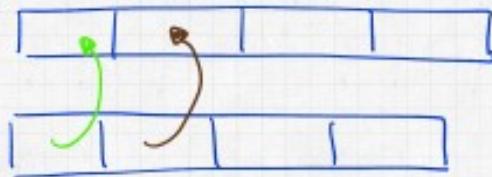
Reparar el servidor.

→ Servidor conoce los clientes mediante este descriptor



Nombre  
codigo  
clientfd[] Sockets

hilo [:] hilos

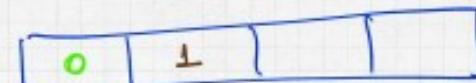


descriptorSocket  
↓

Antes llamado  
descriptorThread

descriptor Socket.  
q' come este hilo

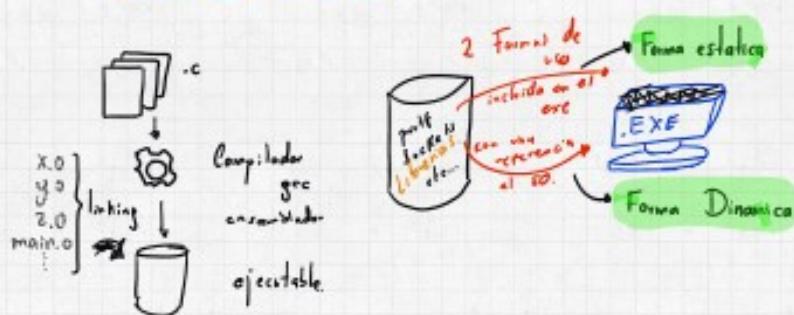
Describe el  
socket que estn  
comiendo 1 hilo  
en ejecuci&on.



## Memoria

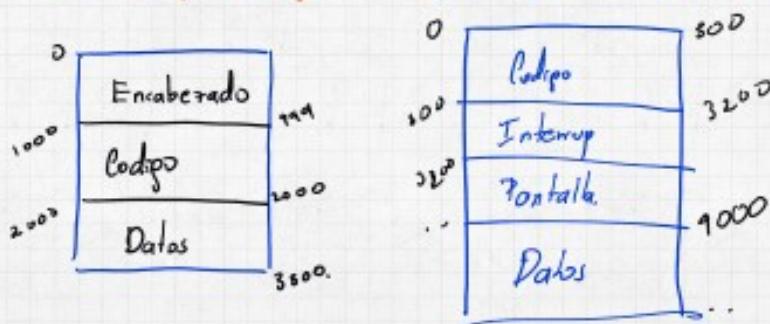
Voice\_15\_1

Almacenamiento masiva



Sistema gestión memoria Maneja los bloques del fichero ejecutable.

Mapa memoria  $\Rightarrow$  Manejado por Gestor Memoria



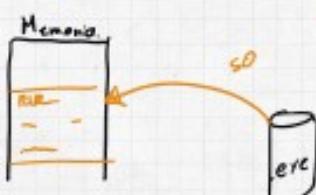
Fichero memoria  $\xrightarrow{\text{copiar}} \text{memoria}$   $\xrightarrow{\text{en memoria}}$  Mapa de memoria para hacer un proceso

Redimensionamiento de bloques memoria depende Gestor de memoria.

→ Cada proceso tiene sus únicos recursos

Reubicación:

Voice\_16\_1



El core generado ó falso de segmentación lo realiza el Gestor de memoria.

- Memoria Compartida

# Memory Management Unit MMU

Es un componente de hardware encargado de:

- Transformación direcciones lógicas a físicas.
- Protección entre el acceso de procesos a páginas.
- Control de las **TP** para manejo rápido de traducciones.

Voice\_17\_1

## Tabla Página

Relaciona la página con el marco donde está contenido

T.P.

—	—
—	—
—	—
—	—
—	—

largo dirección física

Política de asignación de espacio.

El sistema operativo maneja el tamaño del marco.

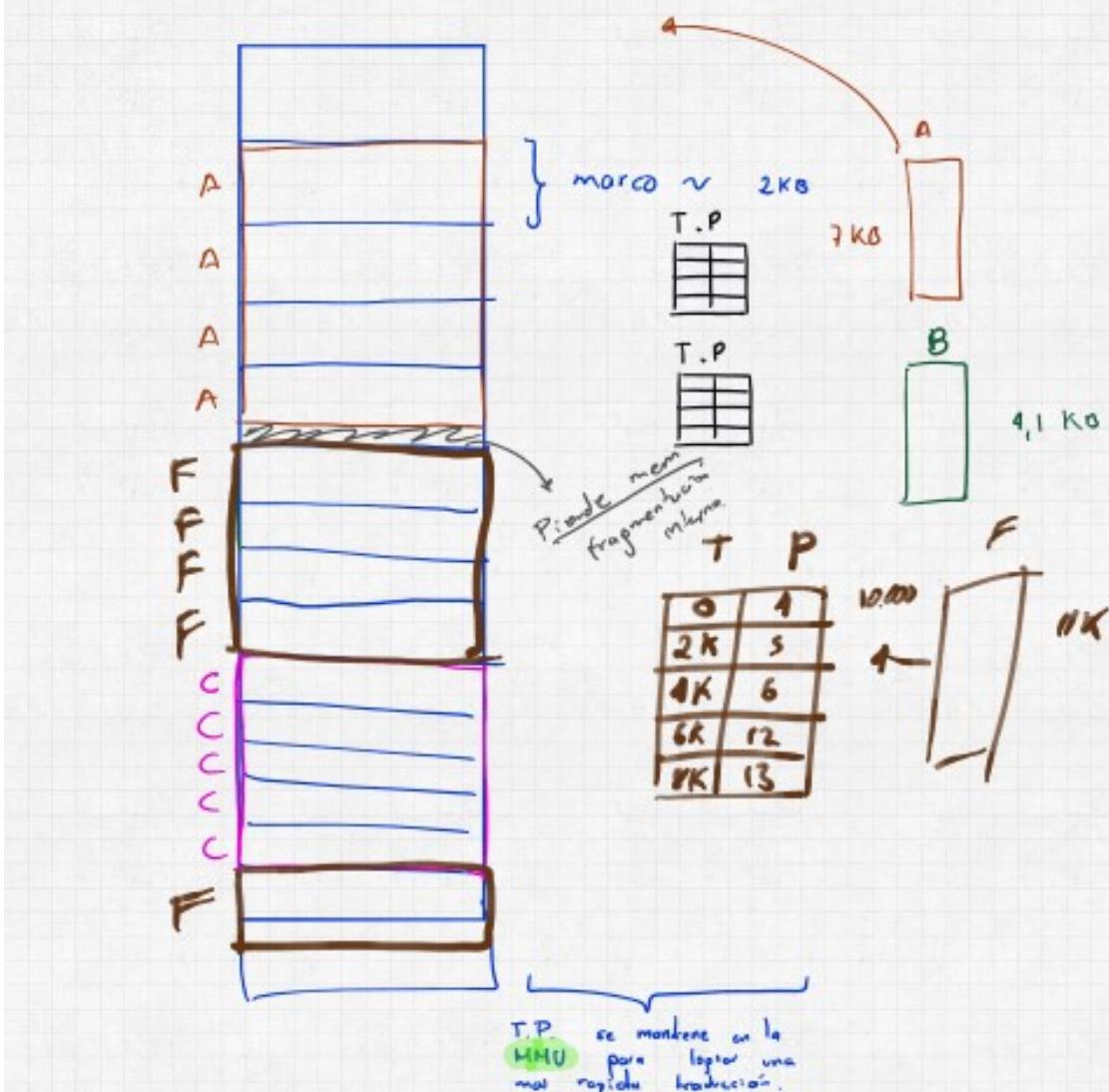
Tipos  
Fragmentación  
en los marcos.

Interno: Se asigna mayor espacio del  
requerido al proceso

Externo: Memoria q' no se puede usar  
entre los marcos.

} Sobre espacio dentro de  
la pag.  
} Sobre espacio entre  
marcos.

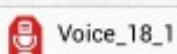
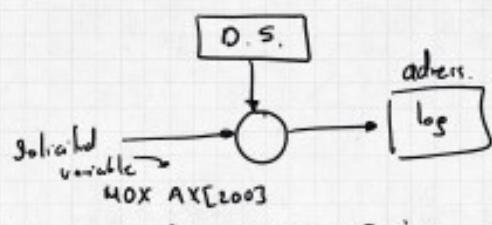
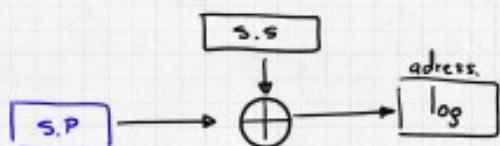
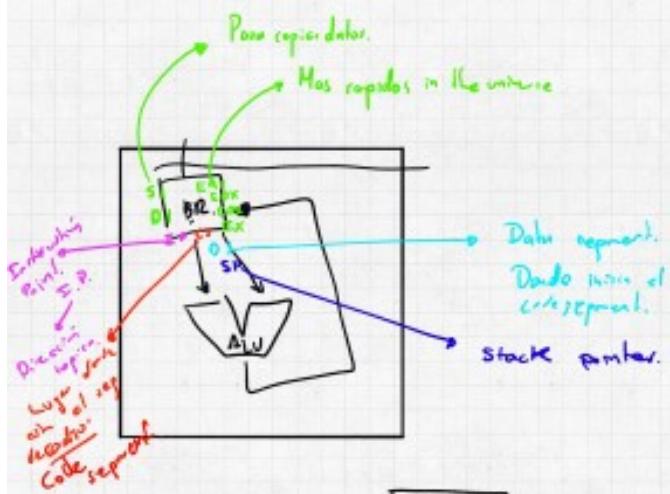
Paginación ← Por demanda





Conocer una dirección física solo se puede con el S.O.

Cuando un dispositivo se conecta se le da una dirección física



Voice\_18\_1 memoria virtual.

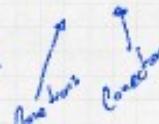
Algoritmo optimo.: Sacar al q' más veces va a usar el pc.



• Los resultados generan páginas

FIFO La q' mas reciente en entrar.

Algoritmo 2 apart / algoritmo reloj.



Algoritmo LRU

Página menos recientemente usada

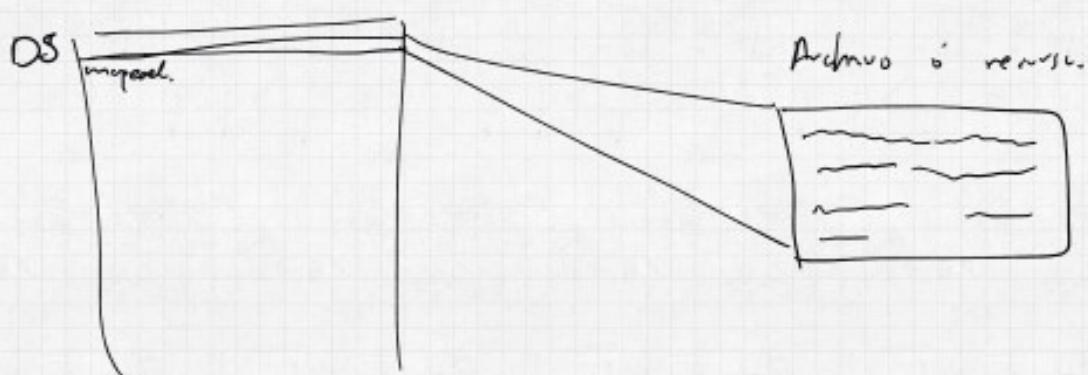
mlock  $\rightarrow$  Bloquear q' en proceso no visto  
1 swap- c/m Hash ó Bases de discursos etc.

Archivos proyectados en memoria.



mmap.

Proceso

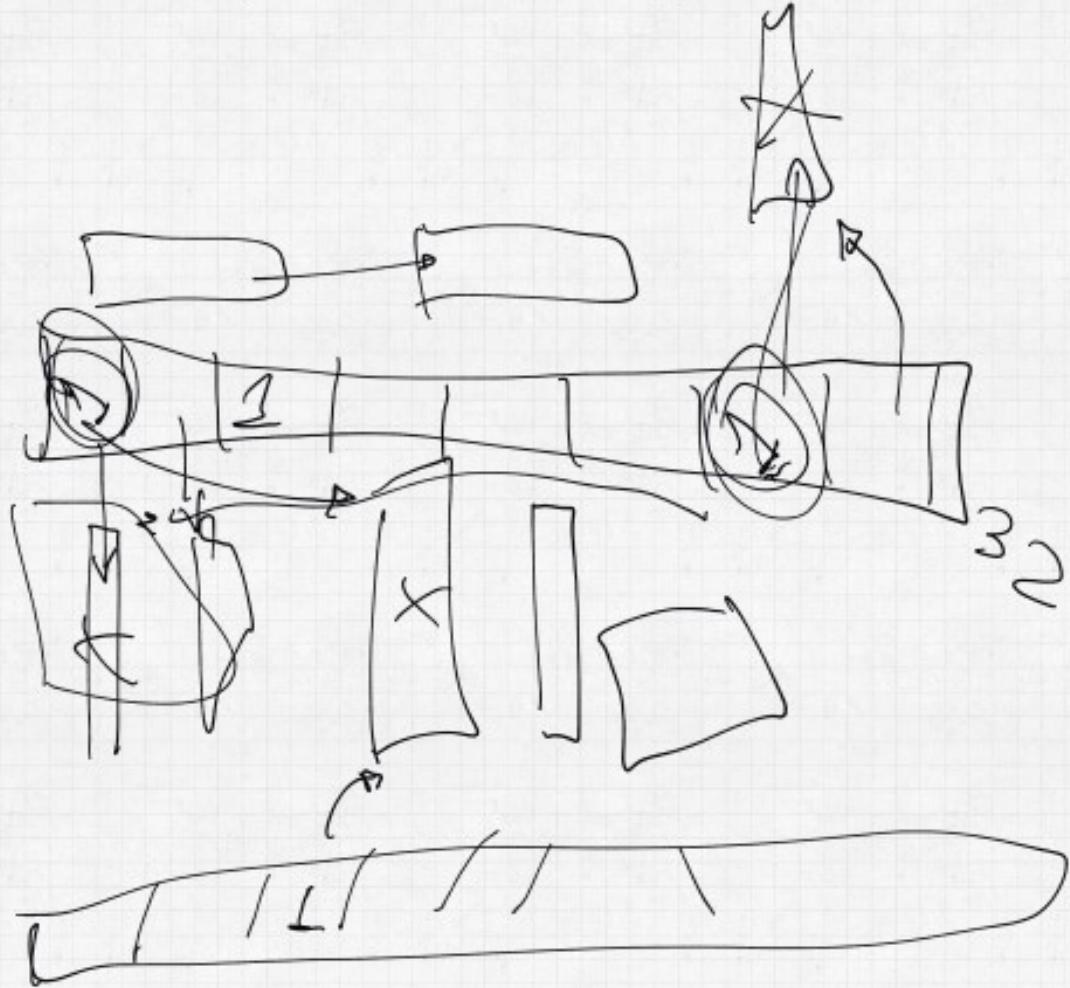
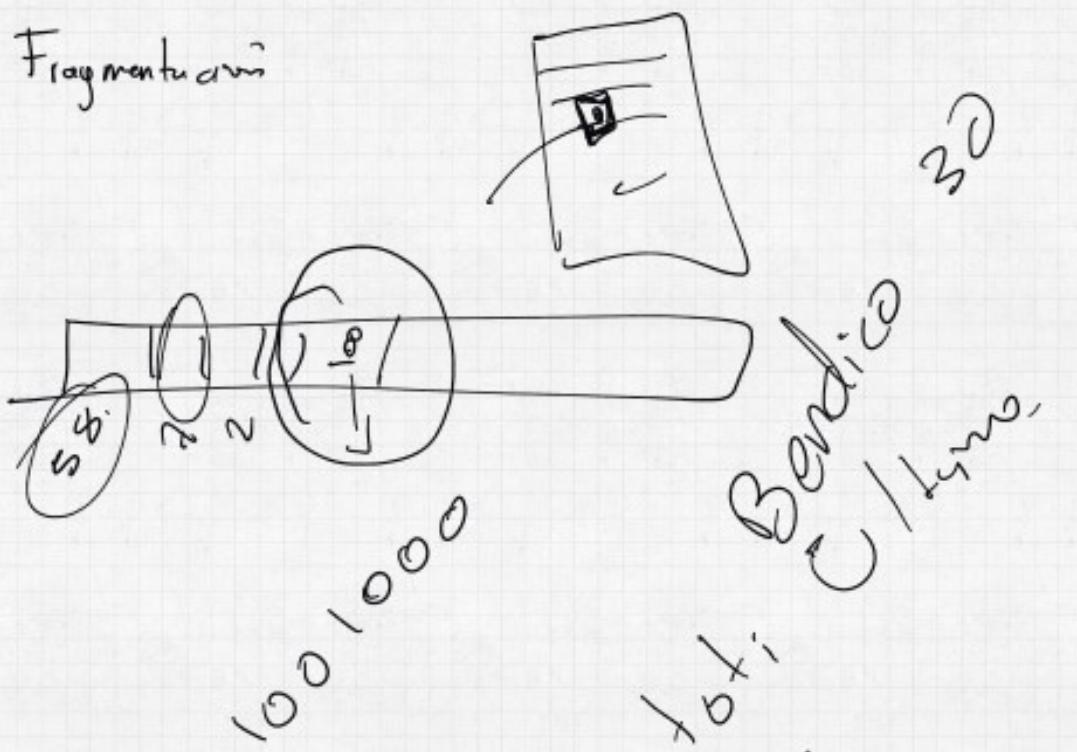


echo

In → en lazo Fig 20

Ren el wG solo en  
realidad.

Fragmentación



## COSAS DEL KERNEL

El Kernel no usa unidad de punto flotante debido a que no todos los dispositivos la tienen.

Kernel Panic = Blue screen  
Linux Windows

NO Swapping :V

El Kernel tiene un log (Para debuggear mucha cosa)  
↓ Home ↑  
dmesg

BFS y CBS



Voice\_20\_1

## Cross Compilation

Se denomina así a la creación de un ejecutable para una plataforma distinta a la que tiene el compilador ejm: compilar un programa en windows para la plataforma android.

Funcionamiento taller concurrente semaphore.

proceso( void \* datos ) {

Bloquea un semáforo

muestra datos -> a  $p$  veces en pantalla

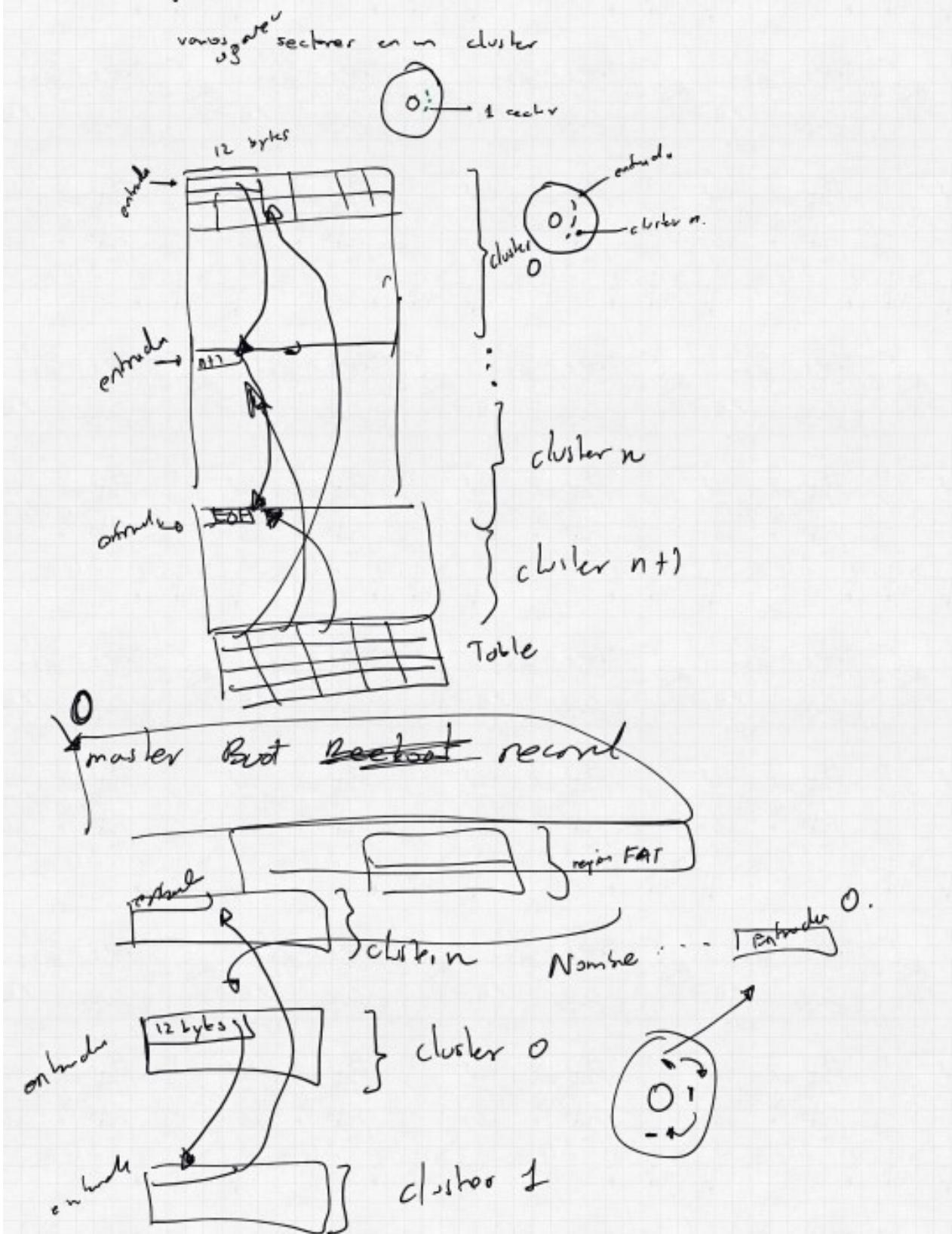
Duerme 1 seg

muestra "-."  $p$  veces en pantalla

Desbloquea el semáforo

}

## Sistema fat



## Ventajas

Se puede recuperar datos si la tabla no tiene datos FAT

Para partitionar menos de 200 MB  
no se requiere una carga.

## Desventajas

- Se desperdicia espacio para datos que pueden ser los clusters, aunque lo que se almacena sea poco
- Si se pierde una tabla en la memoria se pierde todo en cachorro o un directorio.