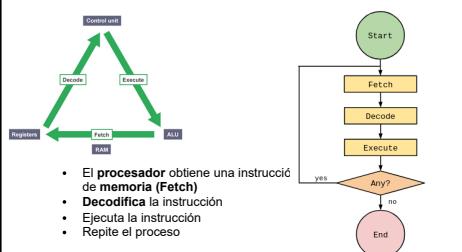


# Cheat Sheet - Sistemas operativos UDEA 2025-1

## Sistemas operativos

### Introducción

Un sistema operativo es un pedazo de software que controla y coordina el uso del hardware entre varias aplicaciones y usuarios. Administra recursos de hardware.



### Virtualización

Un sistema operativo toma un recurso físico como memoria, procesador, y lo transforma en una forma virtual de sí mismo.



Llamadas al sistema: En ocasiones como usuarios se debe decir al SO qué hacer. Se hace mediante System calls.

El SO proporciona la ilusión de tener infinitas CPUs gracias a la virtualización. Cuando se ejecuta un programa, este posee su propia CPU virtual y memoria virtual.

Time sharing: Los procesos están en memoria cierto tiempo, y se ponen en espera para que los demás también se ejecuten. Comparten el HW.

### Operaciones del sistema operativo

El procesador tiene 2 modos:

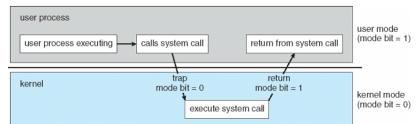
Modo usuario: No privilegiado. Usado por las aplicaciones.

Modo kernel: Privilegiado. Usado por el kernel del SO. Tiene acceso directo al HW.



El HW nos proporciona un bit de modo para saber qué permisos tiene un proceso para ejecutar instrucciones, ya que algunas requieren privilegios y solo podrán ser ejecutadas en modo kernel.

Lo siguiente ocurre cuando se hace una llamada al sistema:



### Procesos

#### Programa vs Proceso

Programa: Entidad pasiva que vive en el disco duro.  
Proceso: Programa en ejecución. Entidad activa.

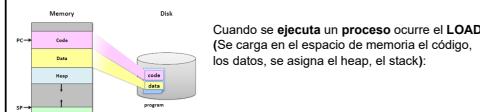


#### Componentes de un proceso:

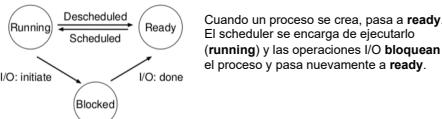
Estado del proceso: Stack pointer, Program counter, GP registers.

Espacio de direcciones: Direcciones y datos.

#### Información I/O



#### Estados de los procesos: Ready, Running y blocked



Una de las estructuras de datos que usa el SO para manejar correctamente los procesos es el PCB (Process Control Block)

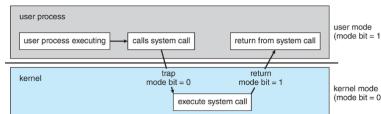
process state
process number
program counter
registers
memory limits
list of open files
...

El PCB guarda información importante de cada proceso. El estado de ejecución, el PC, el estado de los registros de la CPU, información de scheduling de la CPU, información de manejo de memoria, accounting information, información de I/O.

### Ejecución directa limitada (LDE)

Permitir que un programa se ejecute directamente en la CPU. Limitada porque el kernel debe estar aislado de los procesos. Directa porque los programas se ejecutan directamente sin que el SO intervenga ni verifique la validez de las instrucciones.

Si se quiere ejecutar una operación restringida:



Instrucciones Trap: Estas instrucciones permiten cambiar el modo usuario a modo kernel. Cambia el bit de modo de la CPU a 0.

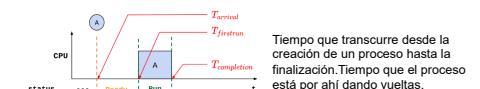
Instrucciones Return from trap: Cambian el bit de modo nuevamente a 1 (modo usuario).

Propuesta no apropiativa: Confía en que los procesos sueltan la CPU por medio de la llamada al sistema yield() o por una excepción.

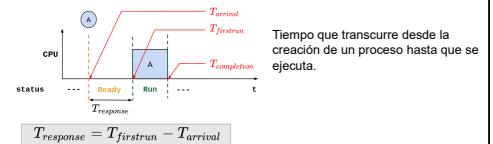
Propuesta apropiativa: Usa un timer para interrumpir los procesos y que el SO vuelve a tomar control de la CPU. Si se desea cambiar de proceso, se ejecuta un CONTEXT SWITCH.

### Métricas de planificación

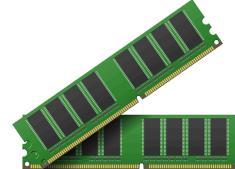
El scheduler es un mecanismo del SO para determinar cuál proceso se ejecutará después del actual. Para implementar diferentes políticas de scheduling necesitamos basarnos en ciertas métricas para determinar cuál es más óptima.



$$Turnaround = T_{completion} - T_{arrival}$$



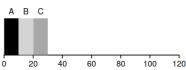
$$T_{response} = T_{firsrun} - T_{arrival}$$



### Políticas de planificación

#### First come, first served

El primero que entra, es completamente atendido, hasta que sale.

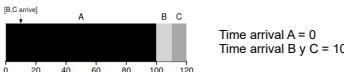


#### Shortest job first

Se completan primero los trabajos más cortos. Esto evita el efecto convoy.

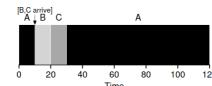


No se puede aprovechar si B y C llegan más tarde que A. No se puede parar el proceso A. No es apropiativo.



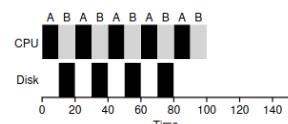
#### Shortest time-to-completion first

Se completan primero los trabajos más cortos. Si se está ejecutando un proceso con un time-to-completion x, y entra justo un proceso con un time-to-completion menor, se para el proceso anterior, para darle paso al más corto.



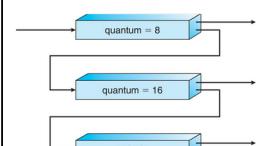
#### Incorporando I/O

La idea es que mientras un proceso esté bloqueado por I/O los demás procesos, se puedan ir ejecutando sin necesidad de que termine completamente el proceso.



### Multilevel feedback queue

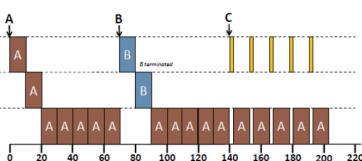
Queremos una política de planificación que soporte los 2 tipos de procesos. La política MLFQ se basa en varias colas de procesos, cada una con un nivel de prioridad diferente. Ahora, la cola de Ready será esta lista de colas.



Reglas:

- Si prioridad(A) > prioridad(B) = prioridad(B): A se ejecuta.
- Si prioridad(A) = prioridad(B): RR para A y B.
- Cuando un trabajo llega al sistema es ubicado en la cola con la prioridad más alta
- Cuando un trabajo consume su asignación de tiempo en un nivel su prioridad es reducida (sin importar las veces que este retome el uso de la CPU).
- Si el trabajo entrega la CPU antes de finalizar su quantum de tiempo, mantiene el mismo nivel de prioridad.
- Después de un tiempo S, mueva todos los trabajos al mayor nivel de prioridad

Ejemplo:

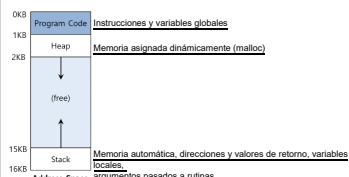


# Cheat Sheet - Sistemas operativos UDEA 2025-1

## Virtualización de memoria y paginación

### Address space

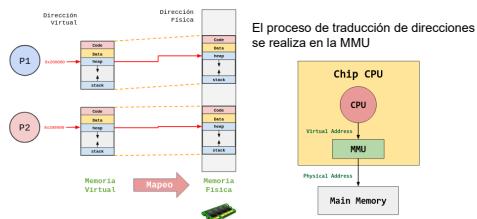
Para virtualizar la memoria, el SO crea una abstracción de la memoria física llamado Address space, o espacio de direcciones. El cual contiene los datos e instrucciones de un proceso en ejecución.



Cada proceso cree que tiene su propia memoria. Por lo que las direcciones virtuales serán diferentes de las físicas. La dirección virtual base de la data de un proceso es la misma que la de los demás. Obviamente en memoria física están en lugares diferentes.

### Address translation

Necesitamos traducir las direcciones virtuales a direcciones físicas para acceder correctamente a los datos que necesitamos.



### Dynamic relocation

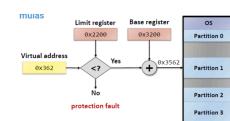
Traducir direcciones virtuales agregando un offset a la dirección base. Esta técnica hace uso de los registros **base** y **bound** del procesador.

Cuando se crea un proceso, el SO decide donde alojarlo en memoria física. La dirección física será:

$$PA = VA + base$$

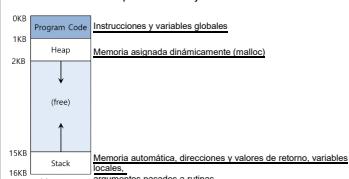
Las direcciones virtuales asociadas al proceso, no deben ser negativas ni tampoco mayor que el registro **bound**.

$$0 \leq VA < bounds$$



### Address space

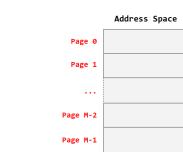
Para virtualizar la memoria, el SO crea una abstracción de la memoria física llamado Address space, o espacio de direcciones. El cual contiene los datos e instrucciones de un proceso en ejecución.



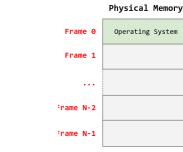
Cada proceso cree que tiene su propia memoria. Por lo que las direcciones virtuales serán diferentes de las físicas. La dirección virtual base de la data de un proceso es la misma que la de los demás. Obviamente en memoria física están en lugares diferentes.

### Paginación

**page:** Bloque de memoria virtual de tamaño fijo. Se puede encontrar por el VPN (Virtual page number)



**frame:** Bloque de memoria física de tamaño fijo que se encuentra asociado a una página. Se identifica con el PFN (Page frame number)



**page table:** Tablas intermedias que permiten hacer el mapeo de direcciones virtuales a direcciones físicas. Cada entrada de la tabla se llama una **page table entry**.

	V	R	M	Prot	PFN
0	1	1	1	00	1
1	1	0	0	01	4
2	0	-	-	-	-
3	1	1	0	01	7

VPN: Índice de la tabla  
V: Bit de validez. Indica si una PTE puede ser usada o no.  
R: Bit de referencia. Es 1 si la página ha sido leída o escrita.  
M: Bit sucio. Si 1 la página ha sido escrita.  
Prot: Bits que indican los permisos de la página.  
PFN: PFN al que está asociado la página con VPN = Índice

### Formato de direcciones Virtuales

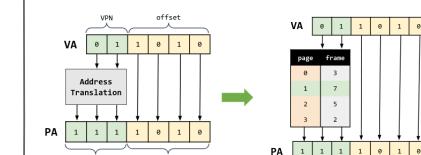


### Físicas



El **offset** indica qué tanto nos tenemos que mover dentro de la página o del frame para llegar al dato que necesitamos

### Esquema general de traducción



### Ejercicio:

Imagina un espacio de direcciones de 64 bytes con páginas de 16 bytes cada una y una memoria física de 128 bytes con frames de 16 bytes cada uno. Suponiendo que tiene la siguiente tabla de página:

page	frame
0	3
1	7
2	5
3	2

Responde, ¿a qué dirección física corresponde la dirección virtual 21?

### Problemas de la paginación



Elevado gasto de memoria porque cada proceso tiene su propia tabla de página.

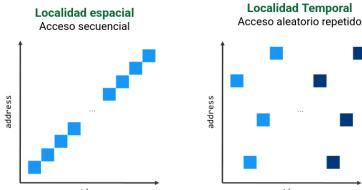
También lentitud en el proceso. Debido a los accesos que se deben hacer a la PT.



### TLB

### Localidad

La localidad se refiere a patrones en el acceso a memoria.



### Introducción

Nace como necesidad de mejorar la traducción de direcciones y subsanar los problemas de la paginación.

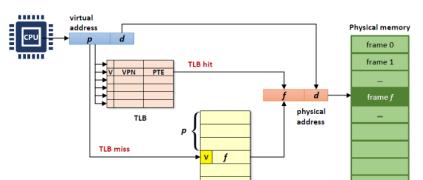
Una TLB es una memoria caché que vive dentro del procesador. Sirve para almacenar las entradas más populares de la PT. El proceso es:

- Se tiene una dirección virtual que se desea traducir.
- Se busca en la TLB para ver si allí se encuentra la entrada requerida.
- Si se encuentra (**HIT**), ya se va directamente a esa dirección física.
- Si no se encuentra (**MISS**), se debe buscar en la PT.

VPN	PFN	Valid	Prot	ASID
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---

- La columna ASID es el ID del proceso al cual pertenece la entrada. Se hace para diferenciar qué entrada pertenece a cada proceso.

### Esquema general de traducción con TLB



Se comparte una TLB para todos los procesos y cada proceso tiene sus entradas.

### Políticas de reemplazo

En el momento en que la TLB se llena por completo, necesitamos tener una política de reemplazo de entradas que maximice la cantidad de hits.

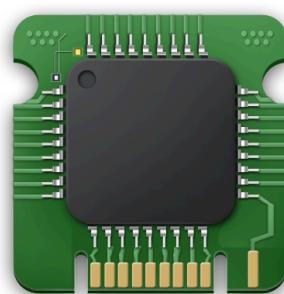
#### LAST RECENTLY USED:

Reemplaza la entrada de la TLB que lleva más tiempo sin ser accedida.

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2
Reference row → 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2
Page Frame → 7 7 7 2 2 4 4 0 0 3 3 3 2 1 2

**RANDOM:** Reemplaza una entrada aleatoria de la TLB por la nueva entrada.

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2
Reference row → 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2
Page Frame → 7 1 2 0 3 2 4 0 0 3 3 2 1 2



# Cheat Sheet - Sistemas operativos UDEA 2025-1

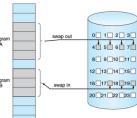
## Espacio SWAP

### Definición

Es un espacio en disco duro empleada para almacenar páginas de memoria. Ya que en algunos programas el AS puede ser más grande que la PM disponible.

El SO divide el espacio SWAP en páginas. El tamaño del SWAP determina la cantidad de páginas que el sistema puede usar en un momento dado.

En el espacio SWAP llamamos a un espacio **Bloque.** (block)



Se realizan 2 operaciones de transferencia entre páginas:  
Swap In, Swap out.

### Reemplazo

Cuando la memoria está llena, se debe tener una política para sacar páginas de memoria.

El SWAP daemon (está de fondo) pendiente del momento en que le toque sacar páginas. Cuando le toca, escoge qué página sacar.

¿Cuando le toca sacar páginas de memoria?: cuando la cantidad de páginas libres es menor que el umbral Low Watermark (LW)

¿Cuando puede dormir el SWAP Daemon?: cuando la cantidad de páginas libres supera el umbral High Watermark (HW)

El swap daemon se activa cuando las páginas se estén asignando normalmente.

### Política de reemplazo óptimo

Se reemplaza la página a la que se accede más lejos en el futuro

Reference row →	0	1	2	0	1	3	0	3	1	2	1
Page Frame →	0	0	0	0	0	0	0	0	0	0	0
M	M	M	M	H	H	M	H	H	H	M	H
Evict →											
Hits: 6	Miss: 5										

$$P_{Hit} = \text{Hits}/(\text{Hits} + \text{Miss})$$

$$P_{Hit} = 6/11 = 0.545$$

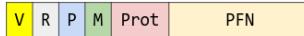
$$P_{Hit} = 54.5\%$$



### Bit de presente

Necesitamos una manera de saber donde se encuentra la página. Si en memoria física o en disco duro.

Este bit se agrega a las entradas de la PTE.



Es 1 si está en memoria física.  
Es 0 si está en el disco duro.

### Traducción de direcciones

TLB				PT		
VPN	PFN	Valid	Prot	PFN	Valid	Prot

V: válido 1 - No válido 0

P: presente memoria 1 - Presente en disco 0

Cuando se consulta la PT y la entrada es válida pero no se encuentra en memoria, se presenta un **fallo de página.**

### Métricas para Swaping policies

Average memory access time (AMAT)

$$AMAT = P_{Hit} * T_M + P_{Miss} * T_D$$

$T_M$  Costo de acceder a memoria

$T_D$  Costo de acceder al disco

$P_{Hit}$  Probabilidad de hit en caché

$P_{Miss}$  Probabilidad de miss en caché

Para el tiempo promedio de acceso a memoria:

$$AMAT = T_M + P_{Miss} * T_D$$

### Política FIFO

El primero que entra es el primero que sale

Reference row →	0	1	2	0	1	3	0	3	1	2	1
Page Frame →	0	0	0	0	0	0	0	0	0	0	0
M	M	M	M	H	H	M	H	H	M	M	H
Evict →											

Hits: 4

Miss: 7

Hit Rate:

$$P_{Hit} = \text{Hits}/(\text{Hits} + \text{Miss})$$

$$P_{Hit} = 4/11 = 0.364$$

$$P_{Hit} = 36.4\%$$

### Políticas de reemplazo

Las políticas definen los criterios para seleccionar cuál página debe salir de memoria.

Política de reemplazo óptimo (Optimal Replacement Policy).

Política de reemplazo FIFO.

Política de reemplazo aleatoria.

Política de reemplazo LRU (Least Recently Used).

### Política random

Se reemplaza una página aleatoria.

Reference row →	0	1	2	0	1	3	0	3	1	2	1
Page Frame →	0	0	0	0	0	0	3	3	3	1	1
M	M	M	M	H	H	M	H	H	M	H	H
Evict →							0	1	2	3	

Hits: 5

Miss: 6

Hit Rate:

$$P_{Hit} = \text{Hits}/(\text{Hits} + \text{Miss})$$

$$P_{Hit} = 5/11 = 0.4545$$

$$P_{Hit} = 45.45\%$$

### LRU

Reemplazar la página menos recientemente usada.

Reference row →	0	1	2	0	1	3	0	3	1	2	1
Page Frame →	0	0	0	0	0	0	0	0	0	2	2
M	M	M	M	H	H	M	H	H	M	H	H
Evict →										0	

Hits: 6

Miss: 5

Hit Rate:

$$P_{Hit} = \text{Hits}/(\text{Hits} + \text{Miss})$$

$$P_{Hit} = 6/11 = 0.545$$

$$P_{Hit} = 54.5\%$$

Realizado por:

Cristian Daniel Muñoz Botero

Jhon Alexander Botero Gómez