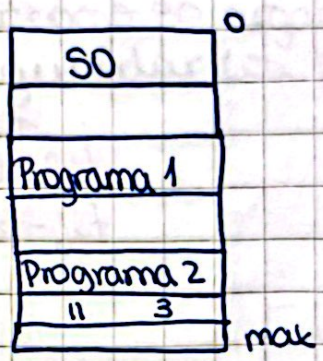
memoria principal:

- tipo de almacenamiento volatil utilizado para almacenar datos y programas en ejecución. la memoria principal es mas rapida que otros tipos de almacenamiento, como los discos duros.

- El sistema operativo utiliza tecnicas de administracion de memoria para optimizar el uso de la memoria principal

- una manera de verlo →



¿Problematica?

¿Que pasa si un programa hace referencia a una direccion de memoria que esta ocupada por otro programa?

Posible solucion:

- localizacion dinamica de valores.
- Proteccion de memoria.

solución:

→ De la problematica anterior nace lo que se denomina como **direcciones variables**

→ La nueva direcion para caso 2 sera 28+16384 = 16412.

→ estas direcciones variables tambien se denominan **direcciones virtuales.**

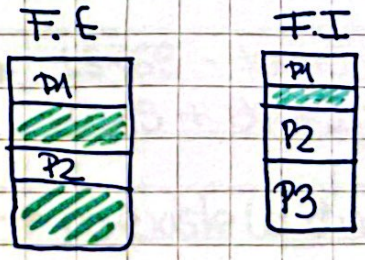
→ la MMU (memory manage Unit) es quien administra la memoria y las direcciones.

- si la memoria se llena, significa que ya no se pueden albergar mas procesos en ram.

→ swap in (HDD → RAM) } existen 2 operaciones de swap.
swap out (RAM → HDD) }

→ existen politicas de organizacion de memoria.
Tambien nos interesa saber como se organiza la memoria principal al momento de guardar los programas. Para esto, existen 4 formas distintas:
- compactacion
- best-fit
- first-fit
- worst-fit

→ FRAGMENTACIÓN:
- La fragmentacion se refiere a la division y el desperdicio de la memoria disponible en pequeños fragmentos no contiguos, lo que se puede llevar a una utilizacion ineficiente de la memoria.
- Existen 2 tipos:
  - Fragmentacion interna: ocurre cuando dentro de un bloque (o segmento) y es el espacio que sobra de este.
  - Fragmentacion externa: ocurre cuando existe espacio disponibles entre bloques.



→ SEGMENTACION:
- Corresponden a los "bloques" (llamados segmentos) en donde se almacenan los procesos que se estan ejecutando en la memoria principal, y su proposito es dividir un proceso en varias partes, ayudando así a solucionar el problema de la fragmentación

EJERCICIO AYUDANTÍA:

Ejercicios segmentación

Fórmulas a utilizar:

1. dirección a calcular − base virtual = offset
2. offset + base física = dirección física

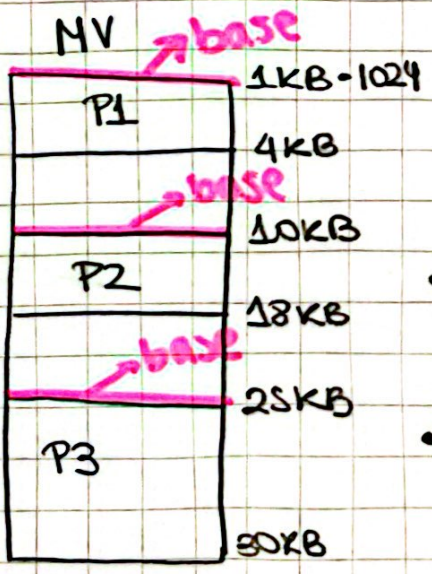
Ejercicio 1:

- Considere un escenario sin segmentación ni paginación. Considere que el tamaño de la memoria de la memoria virtual es de 32KB y de la memoria física de 64KB. Considere (en memoria virtual) que el proceso 1 tiene un tamaño de 4KB base en 1KB; el proceso 2 tamaño 8KB, base en 10KB y el proceso 3 tamaño 5KB y base en 25KB. Además, considere una MMU mantiene la sgte tabla para la memoria física.

Proceso	base
P1	16KB
P2	30KB
P3	3KB

- determine la dirección física para las siguientes direcciones virtuales: 2040B, 13465B, 30850B y 26000B.

Respuesta:

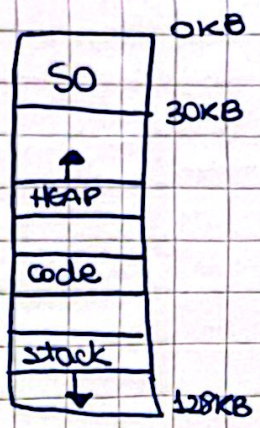


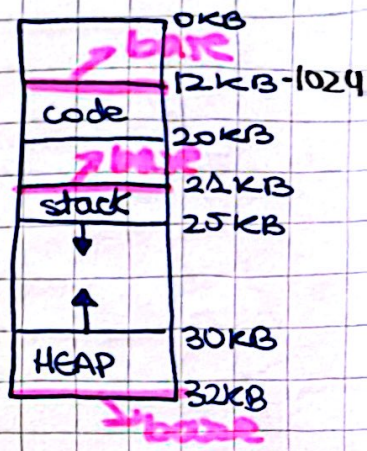
- DV = 2040B → 2040 − 1024 = 1016
  1016 − 16.384 = 17.400 = df
- DV = 13465 → 13465 − 10240 = 3225
  3225 + 30720 = 33945 = df
- DV = 30850 → no existe la dirección ya que no hay base para esto.
- DV = 26000 → 26000 − 25600 = 400
  400 + 3072 = 3472 = df

Ejercicio 2:

segmento	base	size
code	70KB	8KB
heap	50KB	2KB
stack	82KB	4KB

1024





Determinar las direcciones físicas
18000B, 24000B, 31000B

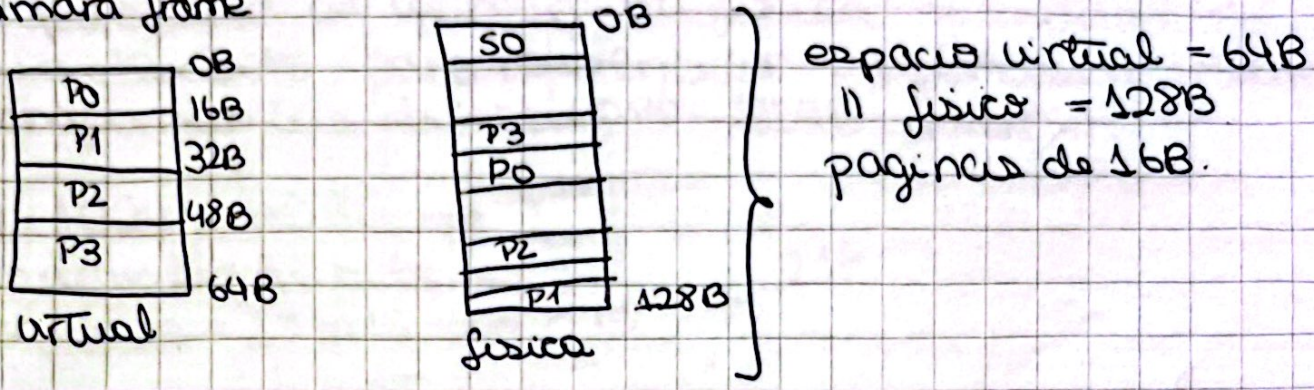
DV = 18000, se encuentra en code
→ 18000 - 12288 = 5712
5712 + 71680 = 77392 = DF

DV = 2400, se encuentra en stack
→ 24000 - 21504 = 2496
2496 + 83968 = 86464 = DF

DV = 31000, se encuentra en heap.
→ 21000 - 32768 = -1768
-1768 + 51200 = 49.432 = DF

Paginación

- dichos segmentos estaran completos y se hablara de paginas.
- Estas paginas se ubicaran en la memoria virtual y tendran un homologo del mismo tamaño en la memoria fisica, la cual se llamara frame



- **politicas de swap:**
  - * Existen 2 tipos de swap
    - → swapin - swapout.

  y consisten en el intercambio de informacion entre la memoria principal y secundaria.

- **PageFault**
  - * ocurren cuando se quiere acceder a un elemento de memoria principal que no se encuentra o no existe en la memoria en dicho instante.
  - * Un pagefault le indica al SO que se tiene que recuperar la informacion que se necesita del HDD y enviarla a memoria principal.
  - * Este proceso deja en espera al programa hasta que se obtenga la información
  - * Por ende, se quiere maximizar los pagefault que puedan ocurrir, con 4 algoritmos, FIFO, random, MIN, LRU.

Ejercicio Repaso control. paginacion nos dan la dirección → $4 \cdot 2^{10} = 2^2 \cdot 2^{10} = 2^{12}$

Paginas de 4KB, virtual 16 bits,
física 15 bits.

1) ¿Capacidad de la memoria física y virtual?
2) ¿Cuantos bits para referenciar paginas y frames?
3) Transformar las direcciones 8196 y 100.

$4 \cdot 1024 = 4096 \rightarrow 2^{12}$
dir virtual = 16 bits $\rightarrow 2^{16}$
dir física = 15 bits $\rightarrow 2^{15}$

entonces para memoria virtual se tendran

0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0

representar pag | offset

y para memoria física

2^{14}															2^0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0

= 24580

$\#pag = \frac{2^{16}}{2^{12}} = 2^4 = 16 \rightarrow$ se representan con 4 bits.

$\#pag\ física = \frac{2^{15}}{2^{12}} = 2^3 = 8 \rightarrow$ se representa con 3 bits

8196 → 0010 0000000000100
representar pag

entonces 0010 → eso es 2 → 110 entonces la dirección física es 24580.

...11: nos dan el espacio!
...da pagina es de 16B, el espacio virtual es 64B y el espacio fisico es de 256B. Transforme la direccion V 49 a su fisica.

- pag 16B
- espacio virtual 64B → $2^6 = 64$ → entonces 6 bits para direccionar el espacio virtual
- espacio fisico 256B → $2^8 = 256$ → entonces 8 bits para direccionar en memoria fisica.

$\frac{64}{16} = 4 = 2^2$ → necesitamos 2 bits para las paginas

entonces 49 pasarlo a binario

49 → 1 1 | 0 0 0 1 = deben ser los mismos para referenciar que lo virtual //

#pag | offsett

entonces

1 1 | 0 0 0 1

↳ esto se usa para representar una pagina y esto en numero es 3 y su frame es 10

entonces paso 10 a binario.

10 → 1010 y mantengo el offset, entonces daria

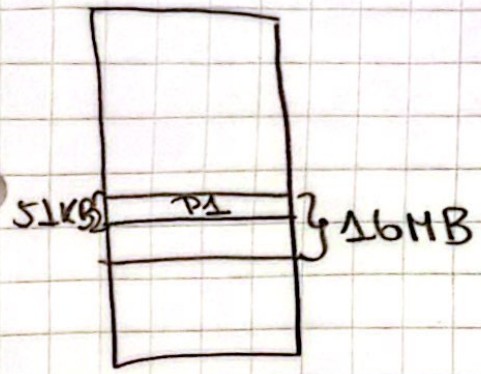
1 0 1 0 0 0 0 1 = 161. //

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

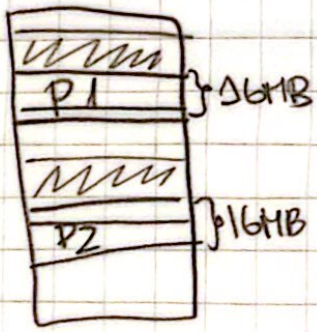
Pag	frame
0	14
1	8
2	3
3	10

2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
65536	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

...ocess p1 requiere de 51KB de memoria principal ...ra su ejecución. Las paginas del sistema que alberga el procesos son 16MB. Sera de bajo este escenario como se presentan ambos tipos de fragmentacion y porque.



- Por ende esto se denomina fragmentación interna, por que se pierde espacio ya que el p1 es menor peso al de la paginas.



- y la fragmentación externa se presenta entre procesos, la cual se dara si llegan mas procesos, se perdera espacio entre las paginas //

Ayudantia n°3
Ejercicios Scheduling
abril 2023

1)Por desgracia, hubo un lamentable desarrollo en todo Chile desde el pasado viernes 18 de Octubre, y que hasta el día de la redacción de esta prueba, sigue en curso. En medio del dantesco escenario que se está llevando a cabo, la Lore recomendó una excelente idea: Que se ayudase a los bomberos indicándoles los plazos de extinción de incendios en ciertos lugares de las inmediaciones. Se han identificado 4 incendios que presentan las siguientes características (tiempo de identificación,duración de extinción, tiempo de plazo de extinción, todos en minutos):

-incendio 1: tiempo id:0,duracion ext:100,plaxo ext: 300
-incendio 2: tiempo id:30,duracion ext:180,plaxo ext: 260
-incendio 3: tiempo id:10,duracion ext:300,plaxo ext: 850
-incendio 4: tiempo id:40,duracion ext:250,plaxo ext: 550

Los bomberos solicitan datos de la planificación correspondientes a tiempos de término, turnaround,inicio de ejecución, y de respuesta. Al respecto:

a) Indique que tipo de algoritmo de scheduling debe usar y porque.

b) Entregue la planificación solicitada.

2) Los profes de SO son ávidos jugadores de consola. Están actualmente con dos consolas: la PiEss Faiv y la Suich. Para cada una de las consolas, los viejos tienen un backlog (les falta por jugar) de P juegos para la PiEss Faiv y S juegos para la Suich.

a) Suponiendo que ambas consolas tienen un backlog idéntico de KH juegos, y si a los profes les interesa bajar ese numero KH lo mas **parejamente** posible para ambas consolas,recomiende un curso de acción. Mencione que sucede con la continuidad en los juegos.

b) Si ahora P=3 y S=2, pero además cada juego tiene una preferencia (que indica que quiere jugar el juego antes si es que tiene una mayor preferencia) por parte de cada uno de los profes, ¿Como cambiaría su curso de acción? Mencione que sucede con la continuidad de juego.

c) Ahora los profes se pondrán a competir **para ver quien termina todos los juegos antes**. Se sabe que P=3 y S=2. Ademas, se tienen los siguientes datos: P1 sale a la venta en 72 horas y tomará 300 horas para terminar, P2 está

disponible ahora mismo y toma 450 horas para terminar y P3 sale a la venta en 24 horas y tomara 200 horas para terminar. De los juegos de Suich, S1 esta disponible ya y toma 150 horas en terminarse y S2 sale a la venta en 48 horas y tomará 50 horas en terminar.

c.1) Recomiende cómo (de acuerdo con qué criterio) deben planificar sus horas de juego a los profesores

c.2) Planifique el tiempo de juego indicando para cada caso cuando (en qué momento) se terminará cada juego, pero también cuanto es el tiempo efectivo en terminar el juego

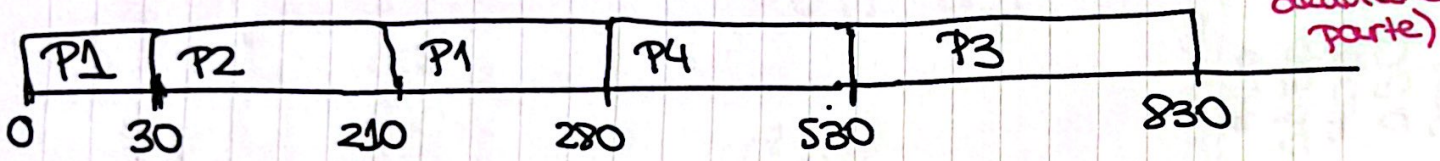
3)En el contexto del 18 de octubre toda la familia de los estudiantes fue mandada a casa por los sucesos venideros de los cierres de campaña del plebiscito y solamente hay un computador en la casa del estudiante, se debe repartir su uso entre todos los miembros de la familia (4 personas en total). Estas personas llegan a casa a las siguientes horas: P1 – 14:00, P2 – 16:00, P3 – 14:30, P4 – 15:00. Ademas, se demoran en usar el PC los siguientes tiempos: P1 – 3horas, P2 – 2horas, P3 – 30mins, P4-2horas. Los respectivos deadlines son P1-20:00, P2-18:30, P3-16:00, P4-23:59. En base a estos datos, indique qué algoritmo debe usar y calcule para todas las personas los siguientes datos: End Time, Turnaround, Execution Start y Response Time

---

1- AYUDANTIA 3:

	arrival time	CPU BURST	end time	turn around	execution star	response time	deadline
P1	0	100	280	280	0	0	300
P2	30	180	210	180	30	0	260
P3	10	300	830	820	530	520	850
P4	40	250	530	290	280	240	550

↓ (menor deadline parte)



[algoritmo a usar deadline earliest first]

↳ se debe usar ya que se debe apagar un incendio primero.

[turnaround = Endtime - arrival time]

[response time = execution star - arrival time] //

a) (cómo es parejamente, se usa round robin).

2- RoundRobin, cada uno juega parejamente hasta que terminen de jugar.
(todos los juegos se interrumpen con el mismo tiempo).

b) ahora se usaría el priority scheduling donde seguiría jugando el juego con la mayor prioridad, y luego que terminan empieza el otro.

c) P = 3
S = 2

datos P1 : arrival time : 72 — CPU Burst : 300
P2 : arrival time : 0 — CPU Burst : 450
P3 : arrival time : 24 — CPU Burst : 200

datos S1 : arrival time : 0 — CPU Burst : 150
S2 : arrival time : 48 — CPU burst : 50

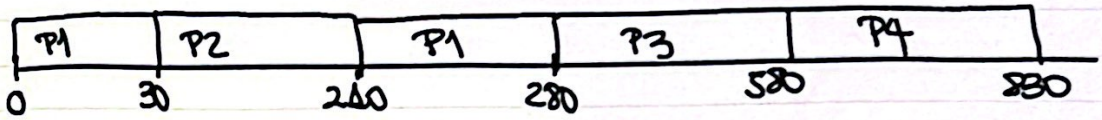
C.1 / C.2)

	Arrival time	CPU Burst	end time	TurnAround
P1	72	300	700	628
P2	0	450	1150	1150
P3	24	200	400	376
S1	0	150	150	150
S2	48	50	200	152

EJERCICIO

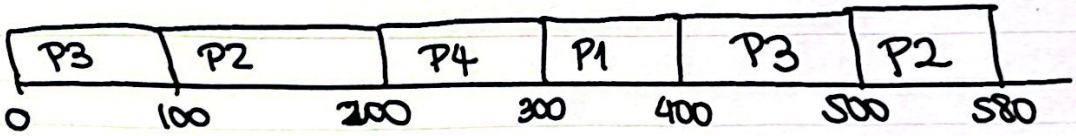
Complete las sgtes tablas: a) Earliest deadline first:

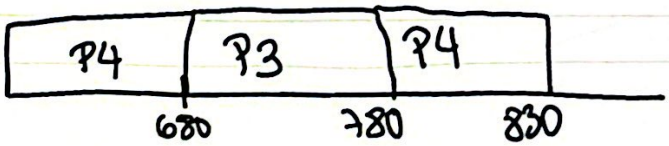
Processor	Arrival Time	CPU Burst	End Time	Turnaround	Execution start	response time	deadline
P1	0	100	280	280	0	0	300
P2	30	180	210	180	30	0	260
P3	10	300	580	570	280	270	590
P4	40	250	830	790	580	540	700



b) round robin (q_t=100)

Processor	Arrival Time	CPU Burst	End Time	Turnaround	Execution start	response time
P1	90	100	400	330	300	230
P2	20	180	580	560	100	80
P3	0	300	780	780	0	0
P4	50	250	830	780	200	150





Sistemas Operativos
Solemne I
Abril 26, 2023

Profesor: Víctor Reyes
Ayudante: Pablo Sáez

---

1. Viene un evento épico, el Monsters of Rock 2023. En dicho evento se presentarán varias bandas de rock old-school como Scorpions, Helloween, Kiss, entre otros. Se cuenta con dos tipos de entrada para los asistentes: cancha (sin enumeración) y galería (con enumeración, en donde el sistema asigna el lugar). Por protocolo de la organización, todo asistente que se encuentre en galería debe estar en su asiento correspondiente. Suponga que usted debe implementar el sistema de compras para dicho evento, en donde M asientos son asignados para galería y N cupos para cancha. Asuma que de manera concurrente podrían existir un número indeterminado de personas interesadas en comprar entradas.

   En base a lo anterior, responda las siguientes preguntas:

   (a) Identifique los elementos del problema y abstráigalos a la terminología que hemos empleado en la asignatura. **(5 puntos)**

   Respuesta: Los threads serían las personas que quieren comprar entradas de cancha o galería. La sección crítica serían los asientos de la galería y la cancha. Utilizar un asiento o un lugar de cancha sería entrar a la sección crítica .

   (b) Escriba snippets de código que permitan implementar una solución al problema. **(10 puntos)**

```c
int asientos_galeria[M_GALERIA] = {0};
int asientos_cancha = 0;

pthread_mutex_t mutex_galeria = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex_cancha = PTHREAD_MUTEX_INITIALIZER;

void* comprar_galeria(void* arg) {
    pthread_mutex_lock(&mutex_galeria);
    for (int i = 0; i < M_GALERIA; i++) {
        if (asientos_galeria[i] == 0) {
            asientos_galeria[i] = 1;
            printf("Asiento asignado a cliente, en el asiento %d\n", i);
            break;
        }
    }
    pthread_mutex_unlock(&mutex_galeria);
}

void* comprar_cancha(void* arg) {
    pthread_mutex_lock(&mutex_cancha);
    if (asientos_cancha < N_CANCHA) {
        asientos_cancha++;
        asiento = asientos_cancha;
        printf("Lugar asignado a cliente en Cancha\n");
    }
```

(1) variables
(2) mutexes
(3) pthread_mutex_lock() & unlock.

```
26      pthread_mutex_unlock(&mutex_cancha);
27  }
```

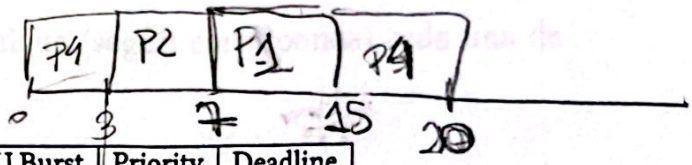
(c) El evento ha tenido tan nivel de éxito, que la organización piensa realizar múltiples conciertos. Cada vez que un concierto quede sin entradas disponibles, se deberá reiniciar todas las compras, es decir, nuevamente quedarán disponibles M entradas de galería y N cupos de cancha. Suponga que automáticamente las compras anteriores quedan almacenadas en una base de datos. Escriba snippets de código que consideren este nuevo requerimiento. **(10 puntos)**

```
1  int asientos_galeria[M_GALERIA] = {0};
2  int asientos_cancha = 0;
3
4  pthread_mutex_t mutex_galeria = PTHREAD_MUTEX_INITIALIZER;
5  pthread_mutex_t mutex_cancha = PTHREAD_MUTEX_INITIALIZER;
6
7  int concierto_actual = 0;
8  int galeria_llena = 0;
9  int cancha_llena = 0;
10
11 void reiniciar_ventas() {
12     pthread_mutex_lock(&mutex_cancha);
13     pthread_mutex_lock(&mutex_galeria);
14     guardar_en_BD();
15     for (int i = 0; i < M_GALERIA; i++) {
16         asientos_galeria[i] = 0;
17     }
18     asientos_cancha = 0;
19     pthread_mutex_unlock(&mutex_galeria);
20     pthread_mutex_unlock(&mutex_cancha);
21 }
22
23 void* comprar_galeria(void* arg) {
24     pthread_mutex_lock(&mutex_galeria);
25     bool encontrado = false;
26     for (int i = 0; i < M_GALERIA; i++) {
27         if (asientos_galeria[i] == 0) {
28             asientos_galeria[i] = 1;
29             printf("Asiento asignado a cliente, en el asiento %d\n" i
       );
30             encontrado = true;
31             break;
32         }
33     }
34     if ((!galeria_llena) && (!encontrado)){
35         galeria_llena = 1;
36     }
37     pthread_mutex_unlock(&mutex_galeria);
38     if (cancha_llena && galeria_llena)
39         reiniciar_ventas();
40 }
41
42 void* comprar_cancha(void* arg) {
43     pthread_mutex_lock(&mutex_cancha);
```

```
44      if (asientos_cancha < N_CANCHA) {
45          asientos_cancha++;
46          asiento = asientos_cancha;
47          printf("Lugar asignado a cliente en Cancha\n");
48      }
49      else{
50          cancha_llena = 1;
51      }
52      pthread_mutex_unlock(&mutex_cancha);
53      if (cancha_llena && galeria_llena)
54          reiniciar_ventas();
55  }
```

2. Scheduling

(a) Dada la siguiente tabla de procesos:



Procesos	Arrival time	CPU Burst	Priority	Deadline
P1	3	8	3	30
P2	1	4	1	25
P3	6	2	5	40
P4	0	3	4	22
P5	4	5	2	23

Determinar el Turnaround, End Time, Execution Start y Response Time utilizando para ello el algoritmo Priority Scheduling y Round-Robin con quantum de 5. **(12 puntos)**

Respuesta: Con Priority Scheduling (no expropiativo):

Procesos	Turnaround	End Time	Execution Start	Response Time
P1	19	22	7	4
P2	6	7	3	2
P3	13	19	17	11
P4	3	3	0	0
P5	13	17	12	8

Con Round Robin (debe ser expropiativo, pues tiene quantum):

Procesos	Turnaround	End Time	Execution Start	Response Time
P1	8	11	3	0
P2	21	22	18	17
P3	7	13	11	5
P4	3	3	0	0
P5	14	18	13	9

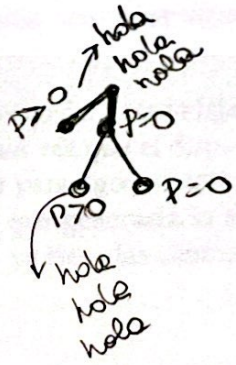
(b) Considere una versión del algoritmo Priority scheduling en donde las prioridades cambian de manera dinámica. Cuando el proceso está en la ready queue, su prioridad cambia a una tasa de α. Cuando está en el estado *running*, su prioridad cambia a una tasa de β. El kernel les asigna una prioridad de 1 a todos los procesos que entran por primera vez a la ready queue. Analice las ventajas y desventajas de un algoritmo que usa un $\alpha > 0$ y $\beta < 0$. **(8 puntos)**

Respuesta: Se analizará desde el punto de vista no expropiativo. Ventajas: Es justo con procesos que han esperado mucho tiempo en la cola, ya que su prioridad aumenta. Favorece a procesos con baja/mediana prioridad. Desventaja: Podría generar inanición para procesos que requieren prioridad, en especial si hay muchos procesos que llegaron antes a la cola.

3. Preguntas cortas, responda y justifique (según corresponda) cada una de ellas de manera clara.

(a) Dado el siguiente código en C:

```
1  int main(){
2      pid_t pid;
3      pid = fork();
4      for(int i = 0 ; i < 3 ; i++){
5          if (pid > 0){
6              printf("hola\n");
7          }
8          else{
9              pid = fork();
10         }
11     }
12 }
```



¿Cuántos procesos se crean en total? Además, indique cuántas veces se imprime la palabra hola **(7 puntos)**

Respuesta: 5 procesos, 6 salidas de hola.

(b) Indique la diferencia entre las pipes con y sin nombre **(4 puntos)**

Respuesta: Con nombre: se utilizan para comunicación entre procesos no relacionados, son persistentes. Las sin nombre se utilizan para procesos relacionados, no son persistentes.

(c) ¿Cuál es la utilidad de la instrucción `compare_and_swap()`? **(4 puntos)**

Respuesta: Al ser una instrucción atómica se usa para implementar exclusión mutua de la SC.

Sistemas Operativos
Solemne II
Junio 16, 2023

Profesor: Víctor Reyes
Ayudante: Pablo Sáez

1. **En las últimas semanas, los profesores de la EIT han estado muy ocupados con labores académicas y administrativas. El profe Víctor no es la excepción, incluso ha llegado al punto de no poder jugar Videojuegos durante semanas. Sin embargo, esto no significa que solo trabaja...pues en su tiempo libre ha estado escuchando las siguientes 6 canciones:**

   1) Valhalla - Blind Guardian
   2) Eagle Fly Free - Helloween
   3) Crimson Thunder - HammerFall
   4) Zeus in Chains - Steve Vai
   5) Dawn of Victory - Rhapsody
   6) Black Star - Yngwie Malmsteen

   La secuencia que utiliza el profe Víctor para escuchar estas canciones sigue el siguiente orden: 3,2,2,2,1,4,6,5,3,1,6,4,3,2,1,6.

   (a) **Suponga que el profe escucha a través de un dispositivo del pasado lejano, en el cuál solo le permite almacenar 3 canciones a la vez. Cada vez que el dispositivo tiene la memoria llena y el profe desea borrar una canción para agregar otra se genera un *MusicFault*. Determine la cantidad de MusicFaults que generaría la secuencia si se utiliza el algoritmo Min. Suponga que el dispositivo ya tiene las canciones 1, 2 y 4 en memoria (5 puntos)**

   Respuesta: 8 MusicFaults

   (b) **Suponga que el profe hace un upgrade del dispositivo anterior, aumentando el espacio a 4 canciones. Determine la cantidad de MusicFaults que se generarían, pero ahora usando el algoritmo LRU. Considere que el dispositivo está sin canciones en memoria. (5 puntos)**

   Respuesta: 8 MusicFaults

   (c) **Suponga que el profe tiene el mismo dispositivo de la pregunta anterior. Determine la cantidad de MusicFaults que se generarían, pero ahora usando el algoritmo del Reloj visto en clases. Considere que el dispositivo está sin canciones en memoria. (6 puntos)**

   Respuesta: 7 MusicFaults

   (d) **Suponga el mismo dispositivo de la parte c), sin canciones en memoria, pero ahora usted puede alterar el orden de la secuencia. Considerando el algoritmo FIFO, ¿Qué secuencia de canciones le recomendaría usted al profe Víctor para que la cantidad de MusicFaults fuera la menor posible?. Además de explicar su solución, determine la cantidad de MusicFaults que se generarían. (5 puntos)**

Respuesta: Para minimizar la cantidad de MusicFaults se debe agrupar por página: 1112222333445666. Esto genera 2 MusicFaults

2. Suponga un sistema en donde el espacio total de las direcciones físicas corresponde a 1KB y el de direcciones virtuales a 2KB. Se sabe además que el tamaño de cada marco es de 64B. La tabla utilizada por la MMU para realizar las transformaciones es la siguiente (asuma que las otras entradas que no aparecen en la tabla implican present bit 0):

Page (Página)	Frame (Marco)
0	12
3	4
9	9
12	3
20	13
24	1

Tomando en consideración lo anterior, determine:

(a) Indique el número de páginas y marcos que podría tener como máximo el sistema. **(4 puntos)**
Respuesta: 32 Páginas y 16 Frames.

(b) Determine el tamaño del offset, tanto para las direcciones virtuales como físicas. **(5 puntos)**
Respuesta: Para ambos son 6 bits.

(c) Suponga que el profe tiene mucho interés en las páginas 9, 12 y 20. Señale la dirección física de inicio y término de cada una de estas páginas. Muestre su desarrollo en términos de números binarios. **(8 puntos)**
Respuesta: Página 9, Inicio: 576 y Término: 639; Página 12, Inicio 192 y término 255; Página 20, Inicio: 832 y Término: 895.

(d) Considere las direcciones virtuales 218 B, 788 B y 1515 B. Para cada una de ellas determine la dirección física correspondiente. Muestre su desarrollo en términos de números binarios. **(8 puntos)**
Respuesta: Para 218 se tendría: 282, 788 le corresponde 212 y 1515 genera page fault.

3. Responda brevemente cada una de las siguientes preguntas **(3.5 puntos c/u)**:

(a) En el contexto de la memoria secundaria ¿En qué se diferencia la asignación enlazada con la indexada?
Respuesta: La asignación enlazada en memoria secundaria utiliza punteros para enlazar los bloques de datos. Por otro lado, la asignación indexada funciona como un conjunto de índices que apuntan a otros bloques.

(b) ¿Cuál es la utilidad de la Table Look-aside Buffer?
Respuesta: La Table Look-aside Buffer es utilizada para acelerar la traducción de direcciones virtuales a direcciones físicas en un sistema de memoria. Almacenando las traducciones más recientes, se evita tener que acceder a la tabla de páginas principal.

(c) Indique 3 soluciones para evitar la fragmentación externa en la memoria RAM.

Respuesta: Compactación, Paginación y Segmentación.

(d) ¿Cuál es la utilidad de un Virtual File System? Además, indique con que elementos trabaja.

Respuesta: Un Virtual File System (VFS) proporciona una capa de abstracción entre el sistema operativo y los sistemas de archivos subyacentes, permitiendo que los programas se comuniquen de manera uniforme con diferentes sistemas de archivos. El VFS trabaja con elementos como vnodos y descriptores.

Universidad Diego Portales
Facultad de Ingeniería y Ciencias

Examen – Parte 2
Sistemas Operativos

Fecha: Viernes, 8 de Julio de 2022 **Tiempo:** 1 hora 30 mins

P1.-) (20 Puntos Total) Razone y opere sobre el siguiente caso

SECICO, la fuente inagotable de dinero está pensando comprar dos bancos: el Banco Seguridad (del que Super Guide es accionista mayoritario) por la gran seguridad de encriptación que ofrece, y el Banco UntilRichard (cuyo dueño es el nunca bien ponderado Transaction Man) por los tiempos de consulta óptimos (y calculados hasta el infinito) en los que incurre al operar. SECICO, al ser una entidad orientada al cliente y su bienestar y riqueza (ha ha ha) quiere asegurar el óptimo servicio a los clientes en cuanto a sus operaciones bancarias.

Para estos efectos, considere que las transacciones se hacen a través de un único portal que dirige hacia cualquiera de los dos servicios (Seguridad o UntilRichard) independiente de que la persona haya sido cliente de ese banco desde el pasado o no (puesto que ahora todo es SECICO).

a) Tomando la analogía con un Sistema Operativo: indique qué elemento del SO debiese ser el portal de SECICO. Asimismo, comente cómo lo implementaría. **(5 ptos)**
R: El portal de SECICO es la Job Queue en la que los procesos activos en el sistema están concentrados y se aprontan a ser ejecutados.
b) Si ahora cada servicio (Seguridad y UntilRichard) actúa independientemente para atender a los clientes, mencione cómo está organizado el sistema siguiendo con la misma analogía del punto anterior:
   i. Describa lo que sería cada servicio. **(3 ptos)**
   R: Cada servicio es una CPU (centro de atención).
   ii. Según qué algoritmo visto en clases conviene calendarizar la atención de clientes. **(3 ptos)**
   R: FCFS, puesto que los clientes van llegando en orden y se atienden a medida que arriban al sistema.
   iii. ¿Qué diferencia presenta este escenario con lo visto en la asignatura en clases? **(3 ptos)**
   R: Se trata de un escenario multicore, puesto que se tienen dos centros de atención en el sistema que son independientes y pueden atender cualquier requerimiento.

c) Usando lo que respondió en la parte anterior, lleve a cabo la calendarización (cálculo de End Time, Turnaround, Execution Start y Response Time) de las siguientes transacciones asumiendo que se mantienen esas condiciones extraordinarias (6 ptos):

Procesos	Arrival time	CPU Burst
Cliente 1	10	200
Cliente 2	0	300
Cliente 3	3	150
Cliente 4	5	300
Cliente 5	0	300
Cliente 6	8	100

R:

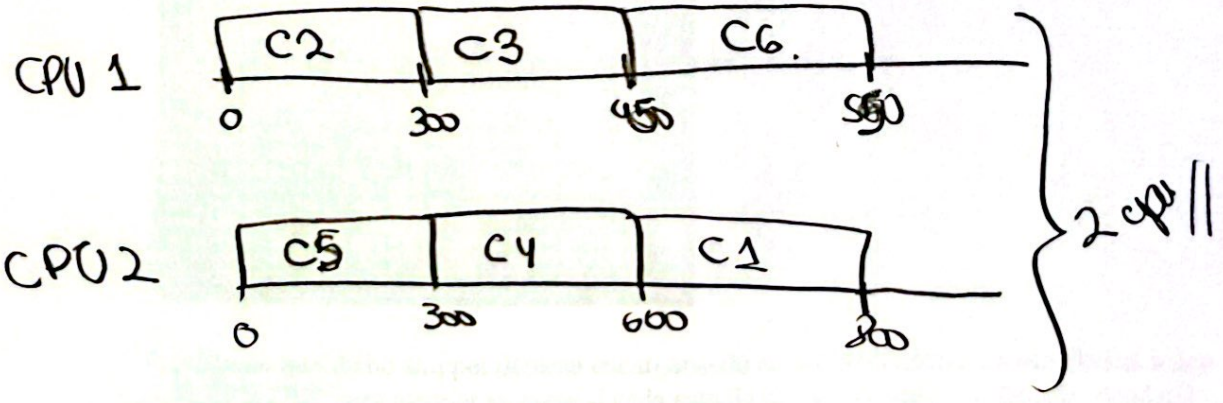
Procesos	Arrival Time	CPU Burst	End Time	Turnaround	Execution start	Response Time
Cliente 1	10	200	800	790	600	590
Cliente 2	0	300	300	300	0	0
Cliente 3	3	150	450	447	300	297
Cliente 4	5	300	600	595	300	295
Cliente 5	0	300	300	300	0	0
Cliente 6	8	100	550	542	450	442

CPU 1	Cliente 2	Cliente 3	Cliente 6
CPU 2	Cliente 5	Cliente 4	Cliente 1

FCFS |.

turnaround = ~~End~~ - Arrival - End

Response = Arrival - Exe



P2.-) (20 Puntos Total) Implemente la solución al problema descrito

Considere el siguiente fragmento (snippet) de código:

```c
#define K 3
#define M 20
#define N 10

pthread_mutex_t acceso[N];
pthread_mutex_t contador;
int i = 0;

void *atender(void *arg)
{
    pthread_mutex_lock(&(acceso[i]));
    printf("Atendiendo clase... \n");
    sleep(5);
    pthread_mutex_unlock(&(acceso[i]));
    pthread_mutex_lock(&(contador));
    i++;
    pthread_mutex_unlock(&(contador));

    return NULL;
}
```

cubiertos por un semaforo

```c
int main(int argc, char *argv[])
{
    pthread_t alumnos[M];
    int j = 0;

    pthread_mutex_init(&(contador), NULL);
    for(j = 0; j < N; j++)
    {
        pthread_mutex_init(&(acceso[j]), NULL);
    }

    for(j = 0; j < M; j++)
    {
        pthread_create(&(alumnos[j]), NULL, atend
    }

    for(j = 0; j < N; j++)
    {
        pthread_join(alumnos[j], NULL);
    }

    for(j = 0; j < N; j++)
    {
        pthread_mutex_destroy(&(acceso[j]));
    }

    return 0;
}
```

Se supone que dicho snippet debiese encargarse de controlar el acceso de estudiantes a una sala en la UDP para atender su clase si cada estudiante es un thread y la función atender()

es el desarrollo y atención de la clase (se asume que los estudiantes llegan a cualquier hora a la clase y se van a cualquier hora de la clase… bueno, no se asume… eso pasa en la realidad. Además, se asume que todos los estudiantes llegan a clase… sí, claro… eso no pasa en la realidad tampoco). Asimismo, suponga que hay un aforo máximo de estudiantes de N para la sala.

Al respecto:

a) Comente en qué falla el código. (3 ptos)
R: El código describe una serie de N mutex lock individuales que guardan el acceso a una región de código. Esto es un problema, porque no asegura que cada uno de los procesos (alumnos) pueda entrar, sino que simplemente usa el lock una sola vez y luego pasa al siguiente.

b) Arregle el código (<u>usando lenguaje C</u>) para que haga lo que debe hacer (7 ptos)
R:

```c
#define K 3
#define M 20
#define N 10

sem_t * entrada;

void *atender(void *arg)
{
    //critical section
    sem_wait(entrada);
    sleep(5);
    sem_post(entrada);
    return NULL;
}

int main(int argc, char *argv[])
{
    pthread_t alumnos[M]; //M > N
    int j = 0;
    entrada = sem_open("entrada", O_CREAT | O_EXCL, 0644, N*K);

    for(j = 0; j < M; j++)
    {
        pthread_create(&(alumnos[j]), NULL, atender, NULL);
    }

    for(j = 0; j < N; j++)
    {
        pthread_join(alumnos[j], NULL);
    }

    sem_close(entrada);
    sem_unlink("entrada");
    return 0;
}
```

c) Discuta ahora acerca de cómo cambiar la función si es que se adapta para la toma de una prueba, en que se cambian/añaden las siguientes restricciones **(10 ptos)**:

i. La prueba empieza a una determinada hora y quien llega posterior a esa hora no puede entrar a rendir la prueba.

ii. Los estudiantes pueden salir en cuanto cada uno finaliza su prueba.

iii. No todos los estudiantes vienen a dar la prueba.

R: Se debe implementar estas restricciones con variables de condición: la restricción i se implementa con un contador que incrementa y al llegar a cierto valor hace un wait. Al revés, para la restricción iii, se hace una variable de condición que empieza en wait y se manda un signal cuando el contador llegó a otro valor (hora de término de la prueba). La restricción ii se cumple automáticamente al implementar i y iii, puesto que las condiciones no consideran cantidad de threads.

Page (Página)	Frame (Marco)
0	12
3	4
9	9
12	3
20	13
24	1

1:1

La tabla utilizada por la MMU para realizar las transformaciones es la siguiente (asuma que las otras entradas que no aparecen en la tabla implican present bit 0):

Tomando en consideración lo anterior, determine:

(a) Indique el número de páginas y marcos que podría tener como máximo el sistema. (**4 puntos**)

(b) Determine el tamaño del offset, tanto para las direcciones virtuales como físicas. (**5 puntos**)

(c) Suponga que el profe tiene mucho interés en las páginas 9, 12 y 20. Señale la dirección física de inicio y término de cada una de estas páginas. Muestre su desarrollo en términos de números binarios. (**8 puntos**)

(d) Considere las direcciones virtuales 218 B, 788 B y 1515 B. Para cada una de ellas determine la dirección física correspondiente. Muestre su desarrollo en términos de números binarios. (**8 puntos**)

3. Responda brevemente cada una de las siguientes preguntas (**3.5 puntos c/u**):

(a) En el contexto de la memoria secundaria ¿En qué se diferencia la asignación enlazada con la indexada?

(b) ¿Cuál es la utilidad de la Table Look-aside Buffer?

(c) Indique 3 soluciones para evitar la fragmentación externa en la memoria RAM.

(d) ¿Cuál es la utilidad de un Virtual File System? Además, indique con que elementos trabaja. VFS

-, 2, 2, 2, 2

-, 2, 3, 3, 3, 6, 6, 6, 1, 1, 1, 4, 4, 5

2.
- se tiene espacio físico = 1KB = 2^{10} B
- se tiene espacio virtual = 2KB = $2^1 \cdot 2^{10}$ B = 2^{11} B
- cada marco es de 64B = 2^6 B

a)

entonces:

1) $\frac{2^{10}}{2^6} = 2^4 = 4$ bits $= 16$ → físico

2) $\frac{2^{11}}{2^6} = 2^5 = 5$ bits. $= 32$ → virtual

$\frac{2^{10}}{2^6} = 2^4 = 16$

$\frac{2^{10}}{2^4} = 2^6 = 64$

$\frac{2^{11}}{2^4} = 2^8 = 256$

Por ende:
espacio virtual se representa con 11 bits → 11
espacio físico se representa con 10 bits → 10
numero de paginas que se puede tener es 32.
numero de marcos que se puede tener es 16.

4/4

b) el tamaño del offset para el espacio virtual y físico seria de 6 bits.
ya que con 4 bits se representa una pagina en virtual, por ende si con los 4 se representa la pag y 10 bits corresponden al espacio virtual, se tiene 6 bits de offset para el virtual.

Pero para el espacio físico, se tienen 11 bits, por ende un marco se representa con 5 bits, lo que implica que el offset sea 6 bits al igual que el virtual.

5/5

0/8

d) 1515B → en espacio virtual sería $2^{10}\,2^9\,2^8\,2^7\,2^6\,2^5\,2^4\,2^3\,2^2\,2^1\,2^0$

1 0 1 1 1 | 1 0 1 0 1 1 → offset

$2^4\,2^3\,2^2\,2^1\,2^0$

10111 → 23 esta pagina no aparece en la tabla, por ende esto indica que no se tiene en la MMU por ende no se podría obtener su dirección física.

8/8

d) 218B → en espacio virtual sería.

$2^{10}\,2^9\,2^8\,2^7\,2^6\,2^5\,2^4\,2^3\,2^2\,2^1\,2^0$

1024 512 256 128 64 32 16 8 4 2 1

0 0 0 1 1 | 0 1 1 0 1 0 → offset.

por ende como el número de pagina es 00011 → 3.

se busca el frame de la pag 3, osea 4. y 4 se pasa a binario. en 4 bits.

4 → 0100 se agrega al offset obtenido anteriormente

01000110 10 → 282 = DF.

888 → en espacio virtual.

$2^{10}\,2^9\,2^8\,2^7\,2^6\,2^5\,2^4\,2^3\,2^2\,2^1\,2^0$

0 1 1 0 0 | 1 0 1 0 0 → offset

8 + 4 = 12

$2^4\,2^3\,2^2\,2^1\,2^0$

01100 → 12 y la pagina 12 en frame es 3. y 3 en binario con 4 bits es 0011

por ende → 0011010100 → 212 = D4