Examen Julio de 2011

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- **Indique su nombre completo y número de cédula en cada hoja (No se corregirán las hojas sin nombre, sin excepciones). Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva y cada parte del problema de teórico en una hoja nueva.**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 30 minutos del examen.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.**

Aprobación

- **Para aprobar el examen se debe tener un mínimo de 60 puntos.**

Finalización

- **El examen dura 4 horas.**

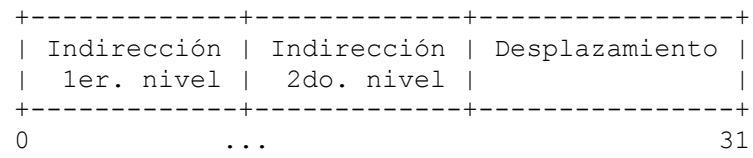
Problema 1 (30 puntos)

Justifique cada una de sus respuestas:

1. Qué entiende por sistemas simétricos?
2. Realice un diagrama de estado de los procesos y explique las transiciones entre ellos para un planificador (*scheduler*) no expropiativo (*non-preemptive*).
3. Qué tareas implica un cambio de contexto (*context switch*) entre procesos?
4. Qué entiende por fragmentación interna y externa de la memoria?
5. Generalmente el espacio de direccionamiento virtual de un proceso (*virtual address space*) es dividido en diferente secciones, cuáles son? y que funcionalidad tiene cada una?
6. En cuanto a diseño qué ventajas brinda que un sistema operativo presente un sistema de archivo virtual (*virtual file system – VFS*)?
7. Describa dos métodos para saber la disposición de los bloques de datos de un archivo en un sistema de archivos.
8. Describa dos servicios que brinda el subsistema de Entrada/Salida.

Problema 2 (35 puntos)

Sea un sistema que implementa memoria virtual con paginación bajo demanda. En donde las direcciones virtuales del sistema son de 32 bits y se realiza una traducción de direcciones a través de dos niveles de tabla de página. Las páginas tienen un tamaño de 4096 Bytes y las direcciones virtuales tienen el siguiente formato:



Responder las siguientes preguntas justificando:

i. Cuál es el tamaño de los marcos (*frames*)? (3 pts.).

ii. Determine el tamaño en bits de cada componente de la dirección virtual teniendo en cuenta que las entradas de la tabla de página son de 16 bits y se desea que la tabla de página de primer nivel ocupe solo una página y completamente. (5 pts.).

iii. Asumiendo que el sistema cuenta con una cache TLB (*Translation Look-aside Buffer*) que está 'limpia' (no contiene 'cacheado' ningún valor) y que no cuenta con memorias cache entre el procesador y la memoria principal, determine la cantidad de accesos a memoria necesarios para leer un arreglo de 10000 caracteres: (12 pts.).

```
Var arreglo : Array[0..9999] of Char; // Tamaño del Char = 1 Byte.

...

for (i = 0; i < 10000; i++)

    ... arreglo[i] ...
```

Notas:

- Asuma que la dirección de comienzo de la variable `arreglo` es al principio de una página.

- No tome en cuenta los accesos para la variable `i`.

iv. El sistema tiene una estrategia de asignación de memoria local y utiliza un algoritmo de reemplazo LRU (*Least Recently Used*). Suponiendo que a un proceso se le asignan 4 marcos para utilizar, muestre mediante un esquema en el tiempo los fallos de páginas y el estado de la memoria si se realizan los siguientes accesos: (15 pts.)

Tiempo	**Dirección virtual**		
	Primer nivel	**Segundo nivel**	**Desplazamiento**
t_1	10	3	2
t_2	10	4	2
t_3	11	3	100
t_4	10	3	520
t_5	12	2	128
t_6	12	2	130
t_7	14	1	768
t_8	11	3	50
t_9	10	3	2
t_{10}	10	4	4
t_{11}	12	2	132

Problema 3 (35 puntos)

Se considera un puente levadizo sobre el cual pueden cruzar varios vehículos al mismo tiempo.
Los barcos que pasan por debajo del puente (siempre y cuando este se encuentre levantado) tendrán prioridad para cruzar respecto a los autos pero no sobre el ómnibus de la selección el cual tendrá prioridad sobre los barcos.
Solamente se admitirá el cruce de un barco por vez y deberá evitarse bajar o subir el puente de forma innecesaria.

Para controlar el puente y el pasaje de vehículos/barcos se dispone de los siguientes procedimientos (sincrónicos):

- `subir_puente()`
- `bajar_puente()`
- `cruzar_puente_vehiculo()`
- `cruzar_puente_barco()`

Se pide implementar usando semáforos.

Nota:

- No se podrán utilizar tareas auxiliares.

Solución:

Problema 2

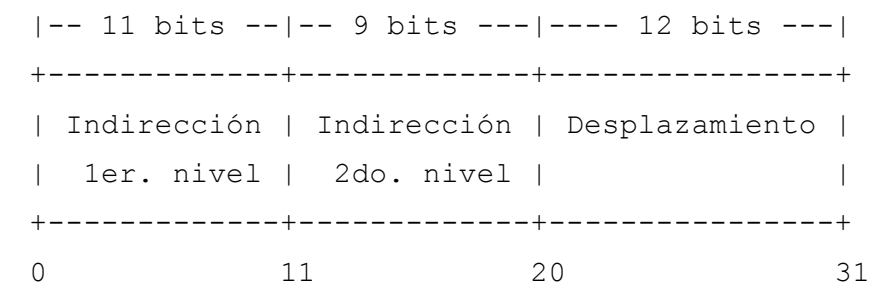
i) El tamaño de los marcos es de 4096 bytes dado que es igual al tamaño de las páginas.

ii) Como el tamaño de la página 4096 bytes entonces se precisan 12 bits para direccionar cada byte. Como el tamaño de la páginai 4096 bytes y las entradas en la tabla de páginas ocupan 16 bits y la tabla de primer nivel ocupa solo una página entonces (tamaño de la página) / (tamaño entradas en la tabla de páginas) = cantidad de entradas en la tabla de paginas

4096 bytes / 16 bits = 4096 bytes / 2 bytes = 2048 cantidad de entradas en la tabla de paginas

Entonces es necesario 11 bites para direccionar las 2048 entradas en la tabla de páginas. Para definir la cantidad de bits para direccionar la entrada en la tabla de páginas de segundo nivel alcanza con calcular la diferencia entre la cantidad de bits de las direcciones virtuales menos los bits de la para direccionar el primer nivel y desplazamiento.

32 bits - 12 bits - 11 bits = 9 bits



iii) Dado que las páginas son de 4096 bytes y queremos acceder a 10000 bytes contiguos, los mismos serán almacenados en 3 páginas contiguas. De esa forma, se tiene un acceso para acceder a la página que contiene la tabla de primer nivel y leer la entrada que corresponde, esta entrada queda almacenado en la TLB. Luego, se realiza un acceso para acceder a la página que contiene la tabla de segundo nivel y leer el valor de la entrada correspondiente, este dato también queda almacenado en la TLB.

A seguir se realizan 4096 accesos para acceder a cada uno de los elemento cargado en la primer página.

Para leer la segunda página es necesario primero acceder a la página que contiene la tabla de segundo nivel y leer el valor de la entrada siguiente, este dato luego de leído queda almacenado en la TLB.

Se realizan otros 4096 accesos para acceder a cada uno de los elemento cargado en la segunda página.

A continuación, se tiene un acceso para acceder a la página que contiene la tabla de segundo nivel y leer la siguiente entrada de la tabla, este dato luego de leído se almacena en la TLB.

Finalmente, se realizan los 1808 accesos para acceder a cada uno de los elementos cargado en la tercera página.

Se obtiene entonces la siguiente cantidad de accesos:

1 + 1 + 4096 + 1 + 4096 + 1 + 1808 = 10004

iv) La página será identifica en el esquema por (entrada en la tabla de primer nivel, entrada en la tabla de segundo nivel)

Estado de los marcos

tiempo	t00	t01	t02	t03	t04	t05	t06	t07	t08	t09	t10	t11
marco1		(10,3)	(10,3)	(10,3)	(10,3)	(10,3)	(10,3)	(10,3)	(10,3)	(10,3)	(10,3)	(10,3)
marco2			(10,4)	(10,4)	(10,4)	(10,4)	(10,4)	(14,1)	(14,1)	(14,1)	(14,1)	(12,2)
marco3				(11,3)	(11,3)	(11,3)	(11,3)	(11,3)	(11,3)	(11,3)	(11,3)	(11,3)
marco4						(12,2)	(12,2)	(12,2)	(12,2)	(12,2)	(10,4)	(10,4)

LRU		FALLO	FALLO	FALLO		FALLO		FALLO			FALLO	FALLO
tiempo	t01	t01	t02	t03	t04	t05	t06	t07	t08	t09	t10	t11
Mas RU		(10,3)	(10,4)	(11,3)	(10,3)	(12,2)	(12,2)	(14,1)	(11,3)	(10,3)	(10,4)	(12,2)
			(10,3)	(10,4)	(11,3)	(10,3)	(10,3)	(12,2)	(14,1)	(11,3)	(10,3)	(10,4)
				(10,3)	(10,4)	(11,3)	(11,3)	(10,3)	(12,2)	(14,1)	(11,3)	(10,3)
Menos RU						(10,4)	(10,4)	(11,3)	(10,3)	(12,2)	(14,1)	(11,3)

Luego de ejecutados los accesos a memoria tenemos que se generan 7 fallos de página y el estado de la memoria es:

en el marco1 queda cargada la página (10,3).

en el marco2 queda cargada la página (12,2).

en el marco3 queda cargada la página (11,3).

en el marco4 queda cargada la página (10,4).

Problema 3

```
boolean puenteArriba;
boolean pasaOmnibus;
integer cantAutos;
integer cantBarcos;
semaforo semBarco;
semaforo semPuente;
semaforo semSincro;
semaforo semPasaOmnibus;
semaforo mutexCantAutos;
semaforo mutexCantBarcos;
semafore mutexPasaOmnibus;

procedure barco()
begin
      P(mutexCantBarcos)
      cantBarcos = cantBarcos + 1
      if(cantBarcos == 1) then
      begin
            P(semSincro); // No permite pasar más autos
            P(semPuente); // Acceso exclusivo a puente
            if(not puenteArriba) then
            begin
                  subir_puente();
                  puenteArriba = true;
            end
      end
      V(mutexCantBarcos)

      P(semBarco); // Solo un barco pasa a la vez
      P(mutexPasaOmnibus)
      if(pasaOmnibus) then // Veo el Ómnibus Celeste quiere pasar
      begin
            V(semPuente); // En ese caso libero acceso a puente
            V(semSincro); // y dejo pasar otros autos mientras el ómnibus
                                                                  pasa
```

```
            P(semPasaOmnibus); // Espero que pase el ómnibus
            P(semSincro); // Ya paso el ómnibus, no dejo pasar mas autos
            P(semPuente); // Reobtengo acceso a puente
            subir_puente(); // El puente estaba bajo, lo vuelvo a subir
            puenteArriba = true;
            pasaOmnibus = false;
            V(semPasaOmnibus); // Permito que llegue otro ómnibus al
                                                                    sistema
      end;
      V(mutexPasaOmnibus)
      cruza_puente_barco();
      V(semBarco); // Permito que otro barco pueda pasar

      P(mutexCantBarcos);
      cantBarcos = cantBarcos - 1;
      if(cantBarcos == 0) // Si no quedan mas barcos por pasar
      begin
            V(semSincro);
            V(semPuente);  // libero acceso a puente
      end;
      V(mutexCantBarcos);
end;


procedure omnibus()
begin
      P(semPasaOmnibus);
      P(mutexPasaOmnibus);
      pasaOmnibus = true;  // Indico que el ómnibus celeste quiere pasar
      V(mutexPasaOmnibus);

      P(mutexCantAutos);
      cantAutos = CantAutos + 1;
      if(cantAutos = 1) then  // Si soy el primer vehículo
      begin
            P(semPuente); // Tomo acceso a puente
            if(puenteArriba) then
```

```
            begin
                  bajar_puente();
                  puenteArriba = false;
            end;
      end;
      V(mutexCantAutos);

      cruzar_puente();

      P(mutexCantAutos)
      cantAutos = CantAutos - 1;
      if(cantAutos == 0) then
      begin // Si soy el ultimo vehículo
            V(semPuente); // libero acceso a puente
      end;
      V(mutexCantAutos)

      V(semPasaOmnibus); // Permito que otro ómnibus llegue o que el
                                          barco esperando pueda continuar
end;

procedure auto()
begin
      P(semSincro);  // Sincronizo autos con barcos. Si hay algún barco
      V(semSincro);  // en el sistema va a tomar el semáforo y por lo
                                          tanto deja bloqueado a los autos
      P(mutexCantAutos);
      cantAutos = cantAutos + 1;
      if(cantAutos == 1) then
      begin // Si soy primer vehiculo
            P(semPuente);  // Tomo acceso a puente
            if(puenteArriba)then
            begin
                  bajar_puente();
                  puenteArriba = false;
            end
      end
```

```
      V(mutexCantAutos);

      cruzar_puente();

      P(mutexCantAutos);
      cantAutos = cantAutos - 1;
      if(cantAutos == 0) then
      begin // Si soy ultimo vehiculo
            V(semPuente); // Libero acceso a puente
      end
      V(mutexCantAutos);
end;

begin
puenteArriba = false;
pasaOmnibus = false;
cantAutos = 0;
cantBarcos = 0;
init(semBarco,1);
init(semPuente,1);
init(semPasaOmnibus,1);
init(semSincro, 1);
init(mutexCantAutos,1);
init(mutexCantBarcos,1);
init(mutexPasaOmnibus,1);
cobegin
      omnibus();
      auto();
      ...
      auto();
      barco();
      ...
      Barco();
coend;
end
```