Universidad Centro Occidental
"Lisandro Alvarado"
Decanato de Ciencias

COORDINACIÓN Y SINCRONIZACIÓN ENTRE PROCESOS

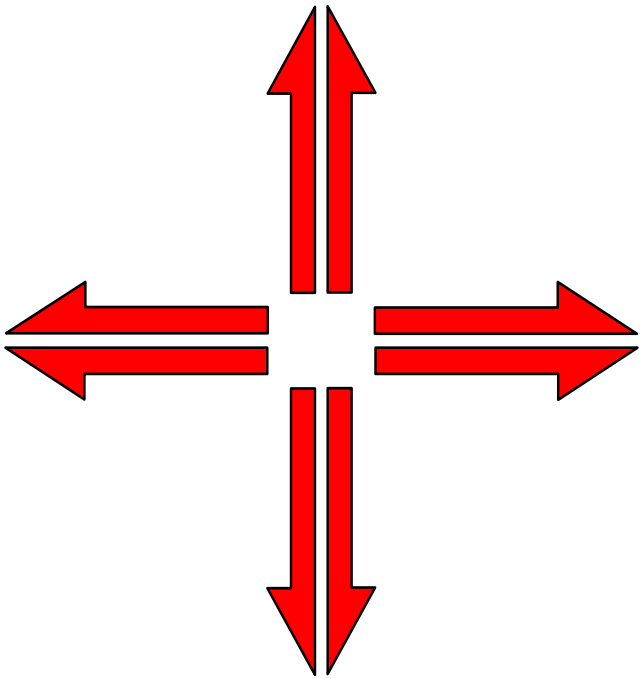
ROCESOS CONCURRENTES

- CONCURRENCIA: Capacidad del sistema operativo de realizar actividades simultáneas

- PROCESOS CONCURRENTES: Si dos o más procesos pueden ser ejecutados por el computador en paralelo, diremos que dichos procesos son concurrentes.

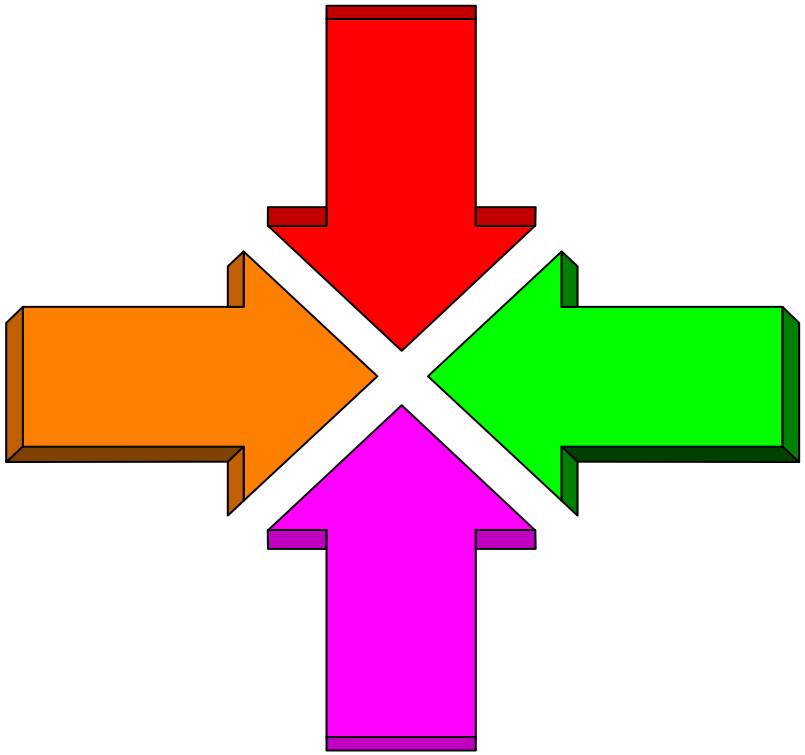
IPOS DE PROCESOS

- **INDEPENDIENTES**



- **COOPERATIVOS:**

Sincronización



STRUCTURA DE CONTROL PARA DICAR CONCURRENCIA

COBEGIN
acción 1;
acción 2;
.
.
acción n
COEND

Conjunto de instrucciones que se ejecutarán en paralelo.

STRUCTURA DE CONTROL PARA DICAR CONCURRENCIA

- Ejemplo 1:Determinar el valor máximo entre A,B,C,D suponiendo que el computador tiene disponible la función MAX (X1,X2)

```
COBEGIN
    M1:=MAX(A,B);
    M2:=MAX(C,D);
COEND
M:=MAX(M1,M2);
```

STRUCTURA DE CONTROL PARA DICAR CONCURRENCIA

- Ejemplo 2: Evaluar la siguiente expresión número ica R:=(A+B)*(C+D)-(E-F)

```
COBEGIN
   R1:=A+B;
   R2:=C+D;
   R3:=E-F;
COEND
R4:=R1*R2;
R5:=R4-R3;
```

STRUCTURA DE CONTROL PARA INDICAR PARALELISMO

- Ejemplo 3:

```
A:=5;
COBEGIN
   WRITE (A);
    A:=0;
COEND
```

ONDICIÓN DE CARRERA

- Cuando se maneja sincronización estándar y se observan actualizaciones al mismo tiempo, en la cual una se va a hacer primero que la otra, se denomina “Condición de Carrera".

RIMITIVAS DE EXCLUSIÓN UTUA

- Se implementan a través de dos instrucciones:
  - ENTRAR_EXCLUSION_MUTUA
  - SALIR_EXCLUSION_MUTUA

```
m Exclusion_Mutua;
      Frecuencia:Integer;
      Ejecucion:boolean;
```

```
lure Observador;

e Ejecucion do
n
bservar_Evento;
NTRAR_EXCLUSION_MUTUA;
Frecuencia:=Frecuencia+1;
ALIR_EXCLUSION_MUTUA
```

```
Procedure Reportero;
 begin
   while Ejecucion do
     begin
     ENTRAR_EXCLUSION_MUTUA;
       write(Frecuencia);
       Frecuencia:=0;
    SALIR_EXCLUSION_MUTUA;
     end
 end;
```

RIMITIVAS DE EXCLUSIÓN UTUA

```
BEGIN
   Frecuencia:=0;
   COBEGIN
        Observador;
        Reportero;
   COEND
END.
```

ECCIÓN CRÍTICA

- **La exclusión mutua permite el manejo adecuado de los procesos concurrentes asíncronos.**
- **Cuando un proceso obtiene acceso a datos y recursos compartidos, se dice que el *conjunto de instrucciones* que los manipulan conforman una SECCIÓN CRÍTICA**

EQUISITOS QUE SE DEBEN CUMPLIR ARA EL MANEJO DE SECCIONES RÍTICAS

Exclusión Mutua:

i un proceso se está ejecutando en su sección crítica, entonces ningún otro proceso se puede estar ejecutando en su sección critica en ese mismo tiempo.

UISITOS QUE SE DEBEN CUMPLIR PARA MANEJO DE SECCIONES CRÍTICAS

Progreso:

ningún proceso se está ejecutando en su sección ítica y existen procesos que desean entrar a sus cciones críticas, entonces sólo aquellos procesos e no se encuentran ejecutando otras actividades, ueden participar en la decisión de cuál será el guiente en entrar en la sección crítica y esta ección no puede postergarse indefinidamente.

REQUISITOS QUE SE DEBEN CUMPLIR PARA EL MANEJO DE SECCIONES CRÍTICAS

- Espera Limitada:

  Debe existir un límite sobre el número de veces que se permite que los demás procesos entren en su sección crítica, después de que un proceso haya efectuado una solicitud de entrar a su sección crítica y antes que la solicitud sea concedida. No se puede posponer a un proceso en espera indefinida para entrar a su sección crítica.

SPERA ACTIVA

- Mientras un proceso se encuentre esperando a que otro proceso termine de culminar las actividades de su sección critica, se dice que el proceso que espera se encuentra en ESPERA ACTIVA.

NCRONIZACIÓN DE ROCESOS

SEMÁFORO:

Es una variable numérica entera de tipo protegida (sólo la maneja el sistema operativo) que se manipula a través de operaciones atómicas bien definidas:

* WAIT = DOWN = P(S): while S < = 0 do;
  S:=S-1;

* SIGNAL = UP = V(S): S:=S+1;

NCRONIZACIÓN DE ROCESOS

Adicionalmente definiremos un procedimiento:

INICIA_SEMAFORO (S,VALOR);

donde se asigna el contenido de valor al semáforo S. (S := VALOR;)

Los semáforos y sus operaciones se encuentran implementadas en el núcleo (KERNEL) del sistema operativo.

NCRONIZACIÓN DE ROCESOS

- TIPOS:
  - Binarios: Toman solamente los valores 0 ó 1.
  - Contadores: Su contenido puede ser mayor de 1.

Usos:

– Para el manejo de las secciones críticas.

– Para el acceso y uso de recursos.

PLEMENTACIÓN DE LAS OPERACIONES BRE SEMÁFORO SIN ESPERA OCIOSA

```
TYPE SEMAFORO = RECORD
                    Valor: Integer;
                    Lista: Lista_de_procesos.
                END;

P(S): S.Valor:=S.Valor-1;
      If S.Valor < 0 then begin
                    Agregar proceso de Semáforo.lista;
                    Bloquear (Proceso)
                end;
```

PLEMENTACIÓN DE LAS OPERACIONES
BRE SEMÁFORO SIN ESPERA OCIOSA

```
):  S.Valor := S.Valor +1;
    If S.Valor <= 0
       then begin
                Remover proceso de Semáforo.Lista;
                Activar (Proceso)
            end;
```

O DE SEMÁFOROS PARA EL MANEJO DE CCIONES CRÍTICAS

```
  semaforo;
Activo:semaforo;
Ejecucion:boolean;
re proceso_uno;

Ejecucion do

dades_preliminares_uno;
ctivo);
ion_critica_uno;
ctivo);
_actividades_uno;
```

```
Procedure proceso_dos;
begin
  while Ejecucion do
    begin
     Actividades_preliminares_dos;
       P(Activo);
       Seccion_critica_dos;
       V(Activo);
       otras_actividades_dos;
    end
end;
```

PLEMENTACIÓN DE LAS OPERACIONES BRE SEMÁFORO SIN ESPERA OCIOSA

```
BEGIN
  IniciaSemaforo(Activo,1);
    COBEGIN
      Proceso_uno;
      Proceso_dos;
    COEND
END.
```

OBLEMA DEL PRODUCTOR Y ONSUMIDOR

```
m Productor_y_consumidor;
tica,vacios,llenos:Semaforo;

ure Productor;
n
hile ejecucion do
begin
 obtener_dato;
 P(Vacios);
 P(Critica);
Introducir_dato_buffer;
 V(Critica);
 V(Llenos);
nd
```

```
Procedure Consumidor;
  begin
    while ejecucion do
      begin
        P(Llenos);
        P(Critica);
        Tomar_dato_buffer;
         V(Critica);
         V(Vacios);
         Utilizar_Dato
      end
  end;
```

OBLEMA DEL PRODUCTOR Y ONSUMIDOR

```
BEGIN (*principal*)
    IniciaSemaforo (critica, 1);
    IniciaSemaforo (vacios, n);
    IniciaSemaforo (llenos, 0);
        COBEGIN
            Productor;
            Consumidor;
        COEND;
END. (*PRODUCTOR_Y_CONSUMIDOR*)
```

OBLEMA DEL BARBERO DORMILON

```
 Barbero_Dormilon;
ero,cliente,critica:semaforo
erando:Integer
ucion:Boolean;
e Barbero;

 ejecucion do
in
clientes);
critica);
perando=Esperando-1;
Barberos);
critica);
rtar_el_pelo;
```

```
procedure cliente;
    begin
      P(critica);
     If esperando<Cantidad_Sillas;
      then begin
                Esperando:=Esperando+1;
                V(clientes);
                V(critica);
                 P(barbero);
                Obtener_corte_de_pelo;
              end
      else V(critica);
```

OBLEMA DEL BARBERO DORMILON

```
BEGIN (*principal*)
   IniciaSemaforo (clientes,0);
   IniciaSemaforo (barbero,0);
   IniciaSemaforo (critica,1);
      COBEGIN
         Barbero;
         Clientes;
      COEND
END.
```

ARDWARE DE SINCRONIZACIÓN

1- TEST_AND_SET

Es una instrucción de hardware especial que permite probar y modificar el contenido de un dato “atómicamente".

```
function Test_And_Set (var valor:boolean):boolean;
        begin
          Test_and_set:=valor;
          Valor:=True;
        end
```

so del Test_and_Set

ntrar:= False;
epeat

While Test_and_Set (Entrar) do ;

SECCIÓN CRÍTICA

Entrar := False;

ntil Ejecución = False

ARDWARE DE SINCRONIZACIÓN

La instrucción SWAP

Es una instrucción de hardware especial que permite intercambiar el contenido de dos datos (o variables) “atómicamente".

```
ocedure SWAP (A,B) ;
var temp:boolean;
begin
  temp:=a;
  a:=b;
  b:=temp
end
```

so del SWAP

```
ntrar:= False;
EPEAT

        Clave:= True;
        Repeat
                SWAP (Entrar, Clave);
        until Clave = false

        SECCIÓN CRÍTICA

        Entrar := False;

NTIL Ejecución = false;
```

ONITORES

- **Es una herramienta de sincronización.**
- **Posee una estructura propia. Es una colección de procedimientos, variables y estructuras de datos que se agrupan en un módulo especial.**
- **Los procesos pueden llamar a los procedimientos de un monitor siempre que lo requieran, pero no pueden accesar directamente las estructuras de datos internas del monitor.**

ONITORES: CARACTERÍSTICAS

1- Sólo un proceso puede estar activo en el monitor en un mismo instante.

2- Los monitores son una construcción de lenguaje de programación.

3- Cuando un proceso X llama a un procedimiento del monitor, las primeras instrucciones del procedimiento chequearán si otro proceso Y está actualmente activo dentro del monitor. Si es así, el proceso llamador X será suspendido hasta que el otro proceso Y abandone el monitor.

4- Las variables locales de un monitor sólo pueden ser accesadas por los procedimientos definidos internamente en el monitor.

5- Libera al programador de la necesidad de exclusión mutua.

ÓMO MANEJAN LOS MONITORES, LA NCRONIZACION DE PROCESOS MANEJO DE OS RECURSOS COMPARTIDOS.

- **1- Uso de variables de condición.**
- **2- Estas variables de condición se accesan con dos operaciones:**
  - *** WAIT (espera)**
  - *** SIGNAL (señalar)**
- **3- La operación WAIT(x) significa que el proceso que la invoca, es suspendido hasta que otro proceso invoque la operación Signal (x).**

Esto significa que la operación SIGNAL(X) "despier-ta" a un proceso suspendido sobre la variable de condición X.

ONITORES: Ejemplo.

or ASIGNADOR_RECURSO;

Recurso_uso:Boolean;
Recurso_libre: Condition;

ure Obtener_Recurso;

If Recurso_Uso
then Wait (Recurso_libre)

Procedure Devolver_Recurso;

Begin
Recurso_uso:= False;
Signal (Recurso-Libre)
end;

N (* Principal *)
urso_uso:= False;

COMUNICACION ENTRE PROCESOS

squemas de comunicación:

Memoria compartida:

- **Los procesos se comunican a través de variables compartidas.**
- **El sistema operativo provee la memoria compartida.**
- **El programador tiene la responsabilidad de proveer comunicación.**

OMUNICACIÓN ENTRE PROCESOS

ii) Sistema de mensajes:

- El sistema operativo tiene la responsa-bilidad de proveer comunicación.

- Operaciones:

*** SEND (enviar)**

*** RECEIVE (recibir)**

Los mensajes pueden ser de tamaño fijo o variable

OMUNICACION ENTRE PROCESOS

ENLACE DE COMUNICACION (LINK)

- Debe existir entre dos procesos que deseen comunicarse.
- Pueden ser:

Físicos: - Memoria Compartida.
- Bus de Hardware.
- Red de Interconexión.

Lógico: Deben definirse e implementarse.

OMUNICACION ENTRE PROCESOS

TIPOS DE COMUNICACION.

A- Directa: - Proceso fuente y proceso destino.

- Operaciones:

SEND (P,Mensaje)

RECEIVE (Q,Mensaje)

- Simétrica: Comunicación uno a uno.

OMUNICACION ENTRE PROCESOS

PROPIEDADES DEL LINK EN LA COMUNICACION DIRECTA

i) Se establece automáticamente entre cada par de procesos comunicantes.

ii) Entre cada par de procesos que se comunican existe sólo un "LINK".

iii) Un Link está asociado exactamente con dos procesos.

iv) El link es bidireccional.

OMUNICACION ENTRE PROCESOS

Ejemplo: Caso Productor - Consumidor.

```
Procedure Productor;                Procedure Consumidor;
  Begin                               Begin
   Repeat                                    Repeat
      .                                       .
      .                                       .
   Producir un_item en next p          Receive(productor,next c)
      .                                       .
   SEND(consumidor,next p)           consumir item desde next c
      .                                       .
   Until False                          Until False
   End;                                End;
```

OMUNICACION ENTRE PROCESOS

B. INDIRECTA: - Los mensajes son enviados y recibidos a través de buzones

- Cada buzón tiene su propia identificación.
- Operaciones: SEND(A,mensaje)
  RECEIVE(A,mensaje)

  donde A es el buzón.
- Asimétrica: Es de uno a varios.

OMUNICACION ENTRE PROCESOS

PROPIEDADES DEL LINK EN LA COMUNICACION INDIRECTA.

i) Se establece entre dos procesos si tiene un buzón compartido.

ii) Un Link puede estar asociado con más de dos procesos.

iii) Entre cada par de procesos que se comunican puede existir más de un Link.

iv) Puede ser unidireccional o bidireccional.

OMUNICACION ENTRE PROCESOS

PROPIETARIOS DE UN BUZON (MAILBOX).

- Un proceso P = Propietario -> P sólo lee de A.
  Q = Usuario -> Q sólo escribe en A

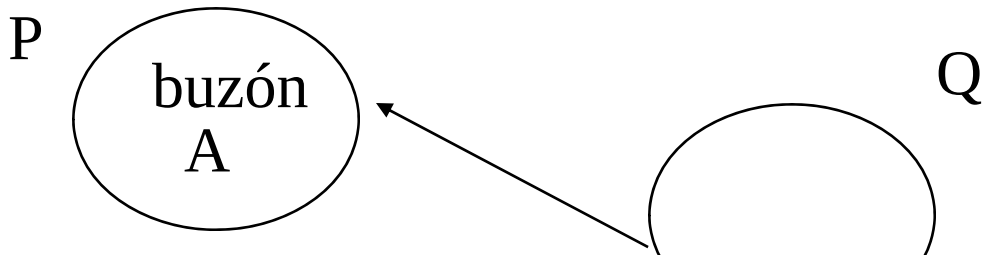
- El sistema Operativo:

  Ofrece mecanismos para:

  i) Crear un nuevo buzón.

  ii) Enviar y recibir mensajes.

  iii) Destruir un buzón.



OMUNICACION ENTRE PROCESOS

IPES:

-Generalmente se usa entre un proceso enviador y un proceso consumidor.

- Los mensajes son de tamaño fijo.

UFFERING:

Se refiere a la cantidad de mensajes que pueden residir temporalmente en “un link”.

Esta propiedad puede verse como una "cola de mensajes” que está relacionada (unida) al link.

OMUNICACION ENTRE PROCESOS

FORMAS (COLAS)

i) Capacidad cero: - El link no puede tener ningún mensaje esperando en el.

ii) Capacidad Limitada: A lo sumo “n“ mensajes pueden residir en la cola del link.

iii) Capacidad Ilimitada: La cola tiene potencialmente longitud infinita.