PARADIGMA

* Paralelismo: Empieza en el mismo lado y acaba en el mismo lado. Durante el mismo instante de tiempo, pero no en el mismo instante (No existe)
* Solapamiento: en intervalos superpuestos. Tiene un inicio y un final. uno puede empezar o acabar antes pero nunca a la vez.
* Simultaneidad: Mismo instante (Es imposible)

Programa concurrente: Dos programas secuenciales solapados instrucción a instrucción, pero nunca 2 instrucciones en el mismo instante

Relaciones entre procesos:

* Independencia: Cuando los dos procesos no tienen relación
* Competencia: Cuando los procesos luchan por recursos limitados
* Cooperación: Cuando dos procesos se ayudan, intentan alcanzar una solución común

Interacción

-Sincronización: Que vayan coordinados para no perder tiempo (Gestión de tiempo)

1. Condicional: Se espera que cumpla una condición Ej: Dejas pasar a alguien o en el pingpong decirle que de tu compañero a la bola

2. Exclusión: Competencia por un recurso común de acceso exclusivo Ej: paso yo antes que tú por mis cojones gordos, o que tu compañero va a darle a la bola directamente

- Comunicación: Compartir datos entre sí (Intercambio de datos)

Estados de un proceso

Preparado -> Ejecución o bloqueado

——> Preparado <————> Ejecución ——>

            \                                        /

             \                                     /

               \ —> Bloqueado < — /

*Gestión:*

Planificación —> Asignación de los diferentes procesos a los procesadores.

Despacho —> Entrega el control de la CPU al proceso

Tanto la planificación como el despacho lo lleva a cabo el **Sistema de Soporte en Tiempo de Ejecución(SSTE)**.

*Planificación:*

El SSTE se activa cuando:

* El proceso termina.
* El proceso se bloquea.
* Finaliza la cuota de tiempo.
* Se produce un error de ejecución.
* Otro proceso de mayor prioridad pasa al estado de preparado.

Determinación del proceso:

* El que más ha esperado.
* El más corto.
* Por prioridad.
* Aleatorio.
* Otro.

**Justicia:** Un planificador es justo si a todo proceso preparado se le asigna en algún momento un procesador para ejecutarse.

El comportamiento funcional de un programa no debe depender del algoritmo de planificación.

**Orden:** Instrucción 1-> Instrucción 2—> Instrucción 3…

El orden no tiene por qué ser total es por ello que haya instrucción total o parcial

Instrucciones atómicas

No se puede dividir su ejecución

De grano fino: Hardware

De grano grueso: Software

PROPIEDADES DE VIDA

Propiedades de seguridad

Deben ser ciertas en cada momento

Exclusión mutua: Como lo de coger el bazooka si lo tengo yo nadie más lo puede coger, este proceso lo tiene el micro y hasta que no lo suelte no lo puede pillar otro

Ausencia de interbloqueo pasivo (deadlock): Se dejan pasar los dos todo el rato por lo que no pasa ninguno

Interbloqueo activo (livelock): Intentan pasar los dos a la vez de golpe por lo que no pueden entrar, la solución es que alguno de los dos ceda el paso

Propiedades de justicia

Justicia débil: Petición continua (estas continuamente pidiendo ejecutar)

Justicia fuerte: Petición indefinida (La pides una vez y ya)

Ausencia de inanición: De vez en cuando le dan si o si el micro

Espera lineal: Antes de darle dos bollos a mi hermano me das a mi uno por medio

Espera FIFO: Se realiza el proceso que este antes en la cola siguiendo el orden, es decir, que se realiza el primero antes que los que entre posteriormente

DESPACHO

Se guarda el valor de todos los registros de la UCP para poder restaurarlo en las mismas condiciones de ejecución

Gdb <nombre del archivo>

Run para arrancar el programa

Where para ver donde esta el problema

Up para subir niveles

Print <nombre de variable> para ver

Break para poner puntos de ruptura

Dekker

Todos los procesos deben acabar dando el turno de ejecucion al otro para evitar

la inanición

si los dos quieren ejecutarse se ejecutará el proceso que tenga el turno

si el primero no quiere ejecutarse cederá su turno al segundo

La idea 1 nos pasamos un boli o tenemos un turno (mirar la variable turno)

La idea 2 miro y si no hay nadie me meto dentro y levanto la mano como que

está ocupado

La idea 3 es decir que voy y si hay alguien me espero con la mano levantada

La idea 4 es decir que voy y si hay alguien le dejas pasar (bajas la mano)

y luego lo vuelves a levantar

Exclusion mutua

PROCESS P;

BEGIN

REPEAT

Preprotocolo; //Me paran

SeccionCritica; //Hace la accion

PostProtocolo; //Me dan permiso

SeccionNoCritica;

FOREVER

END;

//Requisitos:

//Excluxion mutua

//Ausencia de interbloqueo

//Ausencia de retrasos innecesarios

//Ausencia de inanicion, Todo proceso tiene que llegar en algun momento al micro

Justicia Débil

PROGRAM PeticionContinua;

PROCESS Servidor(VAR Peticion : BOOLEAN; VAR x : INTEGER;

BEGIN

WHILE NOT Peticion DO ; (\* ESPERA ACTIVA \*)

WRITELN('Servidor: Peticion atendida, x = ', x);

END;

PROCESS Cliente(VAR Peticion : BOOLEAN; VAR x : INTEGER);

BEGIN

x := 1;

WRITENLN('Cliente: Solicito peticion, x = ', x);

Peticion := TRUE;

END;

VAR

x : INTEGER;

Peticion: BOOLEAN;

BEGIN

x := 0;

Peticion := FALSE;

COBEGIN

Servidor(Peticion, x);

Cliente(Peticion, x);

COEND

END,

Justicia Fuerte

PROCESS Cliente(VAR Peticion : BOOLEAN; VAR x : INTEGER);

BEGIN

REPEAT

Peticion := TRUE;

WRITELN('Solicito Peticion x = ', x);

Peticion := FALSE;

WRITELN('Rescindo Peticion x = ', x);

FOREVER

END;

Productores-Consumidores

Siempre vas a tener camiones saliendo (SIGNAL) por lo que devuelven su ficha

pero a la vez siempre vas a tener consumidores esperando (WAIT) y estos una

vez entren no van a salir nunca por lo que se quedan su ficha, pero también

siempre vas a tener camiones saliendo para tener fichas