Propiedades de seguridad.

Exclusión Mutua: Establece que no pueden intercalarse secuencias de un grupo de procesos.

Ausencia de interbloqueo pasivo (deadlock): Un conjunto de procesos que bloquean esperando un suceso que solo puede producir otro proceso del conjunto.

Ausencia de interbloqueo activo: Un conjunto de procesos que bloquean al intentar ejecutarse al mismo tiempo.

Propiedades de vida

Ausencia de interbloqueo activo (livelock): ejecutar instrucciones inútiles.

Propiedades de justicia: Una petición XXX debe ser eventualmente atendida.

* Justicia Débil: Petición continua
* Justicia fuerte: Petición infinitamente frecuente.

Propiedades de Justicia

* Ausencia de inanición: debe garantizarse que sea atendida eventualmente.
* Espera lineal: debe garantizarse que se le conceda antes que a otro proceso se le conceda dos veces.
* Espera FIFO: debe concedérsela antes que a cualquier otro proceso que la haya pedido posteriormente.

Despacho

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

Instrucciones de grano fino, son aquellas que vienen del hardware.

Instrucciones de grano grueso, provienen del software y es posible interrumpir estas instrucciones.

Sincronización a bajo nivel

Sincronización Condicional

* Justicia Débil: petición continúa.
  + Program PeticionContignua;
    - PROCESS Servidor (VAR Petcion : BOOLEAN; VAR x : INTERGER);
    - BEGIN
      * WHILE NOT Peticion DO ; (\* ESPERA ACTIVA +)
      * WRITELN(‘Servidor: Peticion atendida, x = ‘, x)
    - END;
    - PROCESS Cliente (VAR Peticion : BOOLEAN; VAR x : INTERGER);
    - BEGIN
      * x := 1;
      * WRITELN (‘Cliente: Solicito petición, 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: Petición infinitamente frecuente. El proceso servidor puede ejecutarse siempre con Peticion ;= FALSE.
  + PROCESS Cliente (VAR Peticion : BOOLEAN; VAR x : INTERGER);
    - BEGIN
      * REPEAT
        + Peticion := TRUE;
        + WRITELN (‘Solicito Peticion x = ‘, x);
        + Peticion ;= FALSE;
        + WRITELN (‘Rescindo Peticion x = ‘, x);
      * FOREVER
    - END;
* Idea 1; Funciona por turnos
* Idea 2: Miro y si no hay nadie levanto la mano.
* Idea 3: Levanto la mano y luego miro
* Idea 4: Levanto la mano si no es mi turno, espero a mi turno y cuando toque levanto la mano y entro.

Sincronización alto nivel

Semáforos:

* Contador + lista de procesos en espera.
  + Iniciarl(semáforo, valir)
  + Wait (semáforo): c>0 => Decrementa el contador /c==0 suspende el wait
  + Signal

PIC ç

Mitosis y sustitución.

Un proceso en segundo plano se llama Job (tienen un numero). (Ctrl z/bg) manda a seugndo plano

fg manda a primer plano.

exec family:

To P or not to P

==========================

p: buscar en el path actual.

Respecto de los argumentos

==========================

v: varargs (array de argumentos)

l: lista de parámetros (como el printf)

En entorno (environment)

==========================

Se puede pasar una lista de variables de entorno

e: pasar las variables de entorno