Productor-Consumidor

En este programa usamos 7 variables globales:
- *num_items*: número total de items que se producen o consumen.
- *tam_vector*: tamaño del *buffer*.
- *buffer*: vector (LIFO) que guarda el conjunto de elementos producidos.
- *primera_libre*: puntero que apunta a la primera posición libre del *buffer*.
- *escribir*: semáforo que controla la escritura en el *buffer.*
- *leer*: semáforo que controla la lectura en el *buffer*.
- *mutex*: semáforo que verifica el principio de exclusión mutua.

```
// ---------------------------------------------------------------------
// constantes configurables:

const unsigned
  num_items  = 40 ,    // numero total de items que se producen o consumen
  tam_vector = 10 ;    // tamaño del vector, debe ser menor que el número de items
int buffer[tam_vector];
int *primera_libre=buffer;
sem_t escribir,leer,mutex
;
```

Utilizaremos 3 funciones:
- *retraso_aleatorio*: suspende el programa durante un periodo de tiempo aleatorio.
- *pruducir_dato*: produce un dato.
- *consumir_dato*: consume un dato.

```
// ---------------------------------------------------------------------
// introduce un retraso aleatorio de duración comprendida entre
// 'smin' y 'smax' (dados en segundos)

void retraso_aleatorio( const float smin, const float smax )
{
  static bool primera = true ;
  if ( primera )        // si es la primera vez:
  {  srand(time(NULL)); //   inicializar la semilla del generador
     primera = false ;  //   no repetir la inicialización
  }
  // calcular un número de segundos aleatorio, entre {\ttbf smin} y {\ttbf smax}
  const float tsec = smin+(smax-smin)*((float)random()/(float)RAND_MAX);
  // dormir la hebra (los segundos se pasan a microsegundos, multiplicándos por 1 millón)
  usleep( (useconds_t) (tsec*1000000.0)  );
}

// ---------------------------------------------------------------------
// función que simula la producción de un dato

unsigned producir_dato()
{
  static int contador = 0 ;
  contador = contador + 1 ;
  retraso_aleatorio( 0.1, 0.5 );
  cout << "Productor : dato producido: " << contador << endl << flush ;
  return contador ;
}
// ---------------------------------------------------------------------
// función que simula la consumición de un dato

void consumir_dato( int dato )
{
   retraso_aleatorio( 0.1, 1.5 );
   cout << "Consumidor:                          dato consumido: " << dato << endl << flush ;
}
```

Tendremos 2 funciones que asociaremos a 2 hebras:
- *función_productor*: inserta el dato producido por *producir_dato* y la inserta en el *primera_libre.* Después hacemos que *primera_libre* apunte a la proxima posición de escritura.
- *funcion_consumidor*: consume el dato que esta en la posicion anterior a *primera_libre.* Después hacemos que *primera_libre* apunte a la posición leida.

```
void * funcion_productor( void * )
{
  for( unsigned i = 0 ; i < num_items ; i++ )
  {
    sem_wait(&escribir);

    int dato = producir_dato();

    sem_wait(&mutex);
    *primera_libre=dato;
    primera_libre++;
    sem_post(&mutex);

    cout << "Productor : dato insertado: " << dato << endl << flush ;

    sem_post(&leer);
  }
  return NULL ;
}
// ----------------------------------------------------------------------
// función que ejecuta la hebra del consumidor

void * funcion_consumidor( void * )
{
  for( unsigned i = 0 ; i < num_items ; i++ )
  {

    sem_wait(&leer);

    sem_wait(&mutex);
    int dato=*(primera_libre-1);
    primera_libre--;
    sem_post(&mutex);

    cout << "Consumidor:                          dato extraido : " << dato << endl << flush ;
    consumir_dato( dato );

    sem_post(&escribir);


  }
  return NULL ;
}
//----------------------------------------------------------------------
```

En el main, en primer lugar se declararán las dos hebras que vamos a utilizar:
- *consumidor*: Ejercerá el papel del consumidor.
- *productor*: Ejercerá el papel del productor.

A continuación, se inicializan los semáforos para la correcta sincronizacíon del programa:

```
// ----------------------------------------------------------------------
// función que ejecuta la hebra del consumidor

void * funcion_consumidor( void * )
{
  for( unsigned i = 0 ; i < num_items ; i++ )
  {

    sem_wait(&leer);

    sem_wait(&mutex);
    int dato=*(primera_libre-1);
    primera_libre--;
    sem_post(&mutex);

    cout << "Consumidor:                          dato extraido : " << dato << endl << flush ;
    consumir_dato( dato );

    sem_post(&escribir);


  }
  return NULL ;
}
//----------------------------------------------------------------------

int main()
{
   pthread_t consumidor, productor;

   sem_init(&mutex,0,1);
   sem_init(&escribir,0,tam_vector);
   sem_init(&leer,0,0);

   pthread_create(&productor,NULL,funcion_productor,NULL);
   pthread_create(&consumidor,NULL,funcion_consumidor,NULL);

   cout << "\nFIN";

   pthread_exit(NULL);


   return 0 ;
}
```

- *escribir*: se inicializa a 1, ya que el *productor* debe poder escribir desde el principio.
- *leer*: se inicializa a 0, ya que el *consumidor* debe esperar a que el *productor* escriba algo.
- *mutex*: se inicializa a 1, por defecto.

Finalmente pasamos a la creación de las hebras con la correspondiente asociacion de funciones. A partir de ahora, nuestro programa empezará a consumir y producir datos hasta llegar a los *num_item* producidos/consumidos.