



## **SISTEMAS OPERATIVOS**

### **Práctica 3 MultiHilo**

**Profesor: DANTE DOMIZZI SÁNCHEZ GALLEGOS**

**CATHERINE ALESSANDRA TORRES CHARLES**

### **Grupo 81**

Manuel Manchado Barquero Correo: [100522228@alumnos.uc3m.es](mailto:100522228@alumnos.uc3m.es)

Alejandro Fernández Sánchez Correo: [100522269@alumnos.uc3m.es](mailto:100522269@alumnos.uc3m.es)

# Índice:

<b>1. Descripción del código.....</b>	<b>2</b>
1.1. Análisis y descripción del proceso “Factory_manager” .....	2
1.2. Análisis y descripción de los hilos “Process_manager” .....	4
1.3. Análisis y descripción de la cola circular.....	5
<b>2. Batería de pruebas.....</b>	<b>6</b>
<b>3. Conclusión.....</b>	<b>7</b>

# 1. Descripción del código

## 1.1. Análisis y descripción del proceso “Factory\_manager”

El proceso que ejecuta todo el camino de la función main será el “Factory Manager”, encargado de la lectura y procesamiento de la entrada y creación y sincronización de los hilos “Process manager”.

Nuestro código comienza con las comprobaciones sobre la entrada de la función main para poder ver si tiene el único parámetro requerido, el fichero de entrada. Una vez comprobado se procede a la apertura y lectura del fichero:

- Se lee y almacena el primer número correspondiente al número máximo de hilos que puede generar.
- Se lee y almacena en una estructura el resto de datos organizados en grupos de 3 en 3 para poder manejarlos más fácilmente.

Una vez hemos guardado todos los parámetros de entrada y creado una lista con los id de los hilos de forma ordenada, inicializamos todos los mutex y las condiciones que nos servirán para sincronizar los diferentes hilos.

Comenzamos con la creación de los hilos en un bucle contenido por un bloqueo que solo se liberará cuando todos los hilos se creen, de forma que ninguno de ellos avanza hasta que todos se hayan creado.

Trás la creación de todos los hilos y su posterior liberación pasamos a una sección dedicada a la inicialización en un bucle de los hilos, dividida en dos partes que controlan respectivamente dos mutex. La primera parte corresponde con la asignación del hilo que se debe inicializar gracias a una variable global a la que conforme se realizan iteraciones recibe el id del hilo del turno actual y manda una señal para que los hilos comprueben si le toca. La segunda parte simplemente es una espera realizada con una condición en la que el bucle se detiene hasta que el hilo que se está inicializando en el turno correspondiente mande una señal indicando que ya ha terminado de inicializarse.

Una vez están creados e inicializados, toca ejecutar y recoger cada hilo de forma completa por lo que creamos un último bucle encargado de esta tarea. Su funcionamiento es exactamente igual que el bucle de inicialización previamente explicado, solo que al recibir la señal de que ha terminado el hilo, se ejecuta la función `thread_join` mediante la que se le recoge y comprueba cómo ha finalizado.

Por último procedemos a eliminar todos los mutex y las condiciones utilizadas, indicamos la correcta finalización del proceso y lo finalizamos.

## 1.2. Análisis y descripción de los hilos “Process\_manager”

Los hilos “Process\_manager” creados y sincronizados por el proceso principal, se encargan de crear una cola cíclica sobre la cual se ejecutarán un hilo productor y un hilo consumidor que a su vez debe de crear y sincronizar.

El hilo comienza con un bloqueo producido por un mutex que no es liberado hasta que los demás Process\_manager son creados. Cuando es liberado recibe los parámetros y procede a bloquearse de nuevo debido a una condición.

El bloqueo ocurre dentro de un bucle while de forma que cada vez que reciba una señal debido a un cambio de turno, el hilo compruebe si es su turno y en caso de que no lo sea vuelva a bloquearse. Si es su turno el hilo comienza a inicializar todos los mutex y condiciones que va a utilizar e imprime por pantalla que está listo.

Trás inicializarse se verá bloqueado de nuevo esta vez en espera de que se inicialicen todos los demás hilos. Cuando esto ocurre de nuevo cae en un bucle while con condición que lo obliga a esperar su turno.

Cuando llega su turno el hilo procederá a vaciar la cola circular (Para poder usarla correctamente) y a crear tanto al escritor como al consumidor:

- El escritor se dedica a introducir los valores indicados en la lista.
- El consumidor se dedica a obtener los valores de la lista.

Ambos se sincronizan mutuamente enviándose señales y mediante un mutex inicializado en el padre process\_manager. Mientras la lista tenga espacio el escritor la llenará todo lo posible y en el caso de que se llene, manda la señal al consumidor. Si la lista se llena el consumidor obtendrá un solo valor y volverá a mandar una señal al escritor para que continúe. En el caso de que no se deba escribir más, el escritor mandará una última señal al consumidor que reconocerá de forma que obtendrá todos los valores de la lista restante.

Mientras esto ocurre el process\_manager esperará la recogida de ambos hilos hijo e indicará al proceso factory\_manager que ya ha finalizado mediante una señal.

### 1.3. Análisis y descripción de la cola circular

La cola circular y sus funciones están creadas en un fichero a parte de forma que hay un fichero .h asociado para poder utilizarlas en los process\_manager.

Respecto a la implementación de la cola circular, hemos utilizado la estructura elemento que nos ha sido proporcionada como base de la práctica, pero hemos visto útil crear otra estructura llamada circular\_queue en la cual declaramos variables como:

- Id: Contiene el Id de la cola circular
- Size: Contiene el tamaño máximo de la cola circular
- Count: Contiene el número de elementos de la cola circular
- Head: Contiene la estructura elemento de la cabeza
- Tail: Contiene la estructura elemento de la cola

La información que almacenamos nos sirve para hacer un código mucho más limpio y funciones mucho más sencillas. En cuanto a dichas funciones, son las siguientes:

- Queue\_init: Inicialización de la cola circular. Se asignan los valores correspondientes, pasados como parámetros, a las variables de la estructura.
- Queue\_put: Añadir un elemento a la cola.
- Queue\_get: Coger un elemento de la cola.
- Queue\_empty: Devuelve 1 en el caso de que la cola esté vacía.
- Queue\_full: Devuelve 1 en el caso de que la cola esté llena.
- Queue\_destroy: Elimina la cola.

## 2. Batería de pruebas

**Test 1:** Probamos el error que ocurre cuando pasamos más argumentos y no solo el fichero

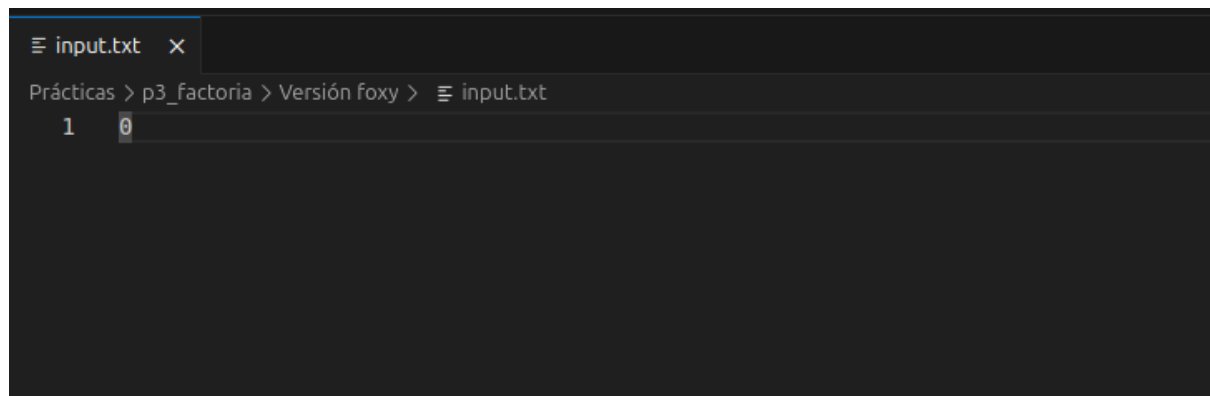
```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt input2.txt  
[ERROR][factory_manager] Invalid file.
```

**Test 2:** Probamos el error que ocurre cuando pasamos un fichero inválido o no existente

```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory no_existe.txt  
[ERROR][factory_manager] Invalid file.
```

**Test 3:** Probamos el error que ocurre cuando como entrada pasamos un número máximo de cintas inválido (0)

- Fichero de entrada:



The screenshot shows a text editor window titled 'input.txt'. The content of the file is the number '0'.

- Salida:

```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt  
[ERROR][factory_manager] Invalid file.
```

**Test 4:** Probamos el error que ocurre cuando en la entrada escribimos más cintas de las máximas permitidas (2 5 5 2 1 2 3 3 5 2)

- Fichero de entrada:



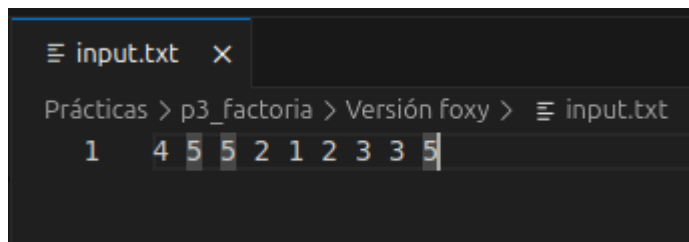
The screenshot shows a text editor window titled 'input.txt'. The content of the file is the sequence of numbers '2 5 5 2 1 2 3 3 5 2'.

- Salida:

```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt  
[ERROR][factory_manager] Invalid file.
```

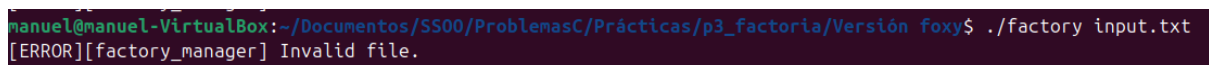
**Test 5:** Probamos el error que ocurre cuando en la entrada escribimos un número de entradas imposible (4 5 5 2 1 2 3 3 5)

- Fichero de entrada:



```
Prácticas > p3_factoria > Versión foxy > input.txt
1 4 5 5 2 1 2 3 3 5
```

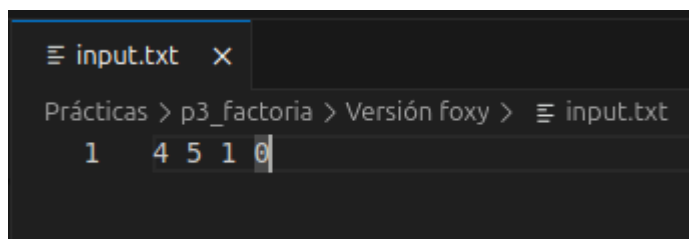
- Salida:



```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[ERROR][factory_manager] Invalid file.
```

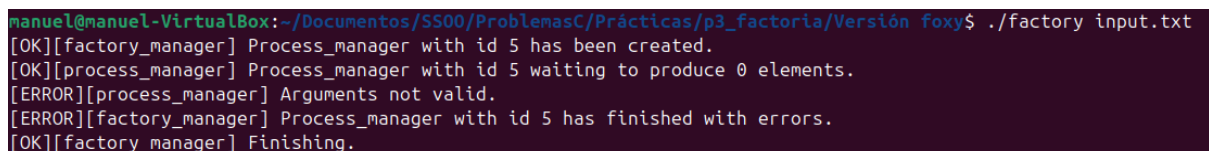
**Test 6:** Probamos el error que ocurre cuando en los atributos de alguna cinta escribimos datos inválidos (4 5 1 0)

- Fichero de entrada:



```
Prácticas > p3_factoria > Versión foxy > input.txt
1 4 5 1 0
```

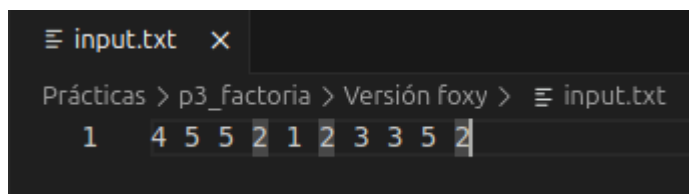
- Salida: Consideramos que el error tiene que ocurrir tras la inicialización del hilo, antes de llegar a ejecutarse.



```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 5 has been created.
[OK][process_manager] Process_manager with id 5 waiting to produce 0 elements.
[ERROR][process_manager] Arguments not valid.
[ERROR][factory_manager] Process_manager with id 5 has finished with errors.
[OK][factory_manager] Finishing.
```

**Test 7:** Vamos a probar un caso válido estándar de entrada (4 5 5 2 1 2 3 3 5 2)

- Fichero de entrada:



```
Prácticas > p3_factoria > Versión foxy > input.txt
1 4 5 5 2 1 2 3 3 5 2
```



- Salida:

```
manuel@manuel-VirtualBox: ~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 5 has been created.
[OK][factory_manager] Process_manager with id 1 has been created.
[OK][factory_manager] Process_manager with id 3 has been created.
[OK][process_manager] Process_manager with id 5 waiting to produce 2 elements.
[OK][process_manager] Process_manager with id 1 waiting to produce 3 elements.
[OK][process_manager] Process_manager with id 3 waiting to produce 2 elements.
[OK][process_manager] Belt with id 5 has been created with a maximum of 5 elements.
[OK][queue] Introduced element with id 0 in belt 5.
[OK][queue] Introduced element with id 1 in belt 5.
[OK][queue] Obtained element with id 0 in belt 5.
[OK][queue] Obtained element with id 1 in belt 5.
[OK][process_manager] Process_manager with id 5 has produced 2 elements.
[OK][factory_manager] Process_manager with id 5 has finished.
[OK][process_manager] Belt with id 1 has been created with a maximum of 2 elements.
[OK][queue] Introduced element with id 0 in belt 1.
[OK][queue] Introduced element with id 1 in belt 1.
[OK][queue] Obtained element with id 0 in belt 1.
[OK][queue] Introduced element with id 2 in belt 1.
[OK][queue] Obtained element with id 1 in belt 1.
[OK][queue] Obtained element with id 2 in belt 1.
[OK][process_manager] Process_manager with id 1 has produced 3 elements.
[OK][factory_manager] Process_manager with id 1 has finished.
[OK][process_manager] Belt with id 3 has been created with a maximum of 5 elements.
[OK][queue] Introduced element with id 0 in belt 3.
[OK][queue] Introduced element with id 1 in belt 3.
[OK][queue] Obtained element with id 0 in belt 3.
[OK][queue] Obtained element with id 1 in belt 3.
[OK][process_manager] Process_manager with id 3 has produced 2 elements.
[OK][factory_manager] Process_manager with id 3 has finished.
[OK][factory_manager] Finishing.
```

**Test 8:** Vamos a probar un caso válido con entradas de dos y tres dígitos (4 10 5 2 111 2 3)

- Fichero de entrada:



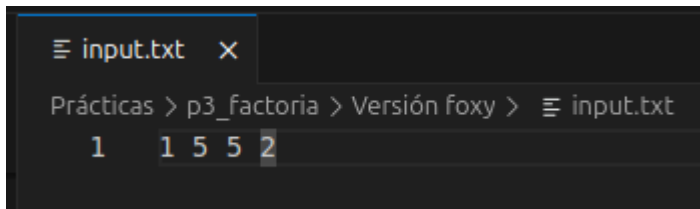
```
input.txt
Prácticas > p3_factoria > Versión foxy > input.txt
1 4 10 5 2 111 2 3
```

- Salida:

```
manuel@manuel-VirtualBox: ~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 10 has been created.
[OK][factory_manager] Process_manager with id 111 has been created.
[OK][process_manager] Process_manager with id 10 waiting to produce 2 elements.
[OK][process_manager] Process_manager with id 111 waiting to produce 3 elements.
[OK][process_manager] Belt with id 10 has been created with a maximum of 5 elements.
[OK][queue] Introduced element with id 0 in belt 10.
[OK][queue] Introduced element with id 1 in belt 10.
[OK][queue] Obtained element with id 0 in belt 10.
[OK][queue] Obtained element with id 1 in belt 10.
[OK][process_manager] Process_manager with id 10 has produced 2 elements.
[OK][factory_manager] Process_manager with id 10 has finished.
[OK][process_manager] Belt with id 111 has been created with a maximum of 2 elements.
[OK][queue] Introduced element with id 0 in belt 111.
[OK][queue] Introduced element with id 1 in belt 111.
[OK][queue] Obtained element with id 0 in belt 111.
[OK][queue] Introduced element with id 2 in belt 111.
[OK][queue] Obtained element with id 1 in belt 111.
[OK][queue] Obtained element with id 2 in belt 111.
[OK][process_manager] Process_manager with id 111 has produced 3 elements.
[OK][factory_manager] Process_manager with id 111 has finished.
[OK][factory_manager] Finishing.
```

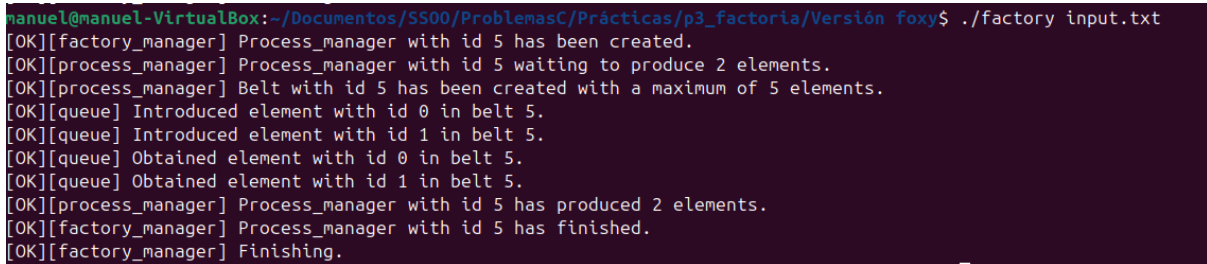
**Test 9:** Vamos a probar un caso válido con el mínimo de cintas (1 5 5 2)

- Fichero de entrada:



```
input.txt x
Prácticas > p3_factoria > Versión foxy > input.txt
1 1 5 5 2
```

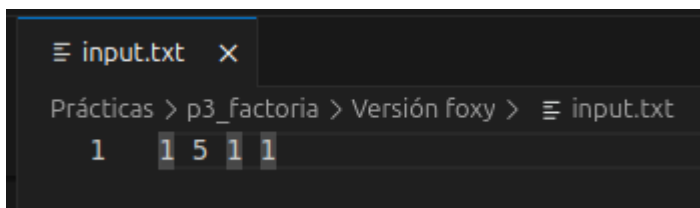
- Salida:



```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 5 has been created.
[OK][process_manager] Process_manager with id 5 waiting to produce 2 elements.
[OK][process_manager] Belt with id 5 has been created with a maximum of 5 elements.
[OK][queue] Introduced element with id 0 in belt 5.
[OK][queue] Introduced element with id 1 in belt 5.
[OK][queue] Obtained element with id 0 in belt 5.
[OK][queue] Obtained element with id 1 in belt 5.
[OK][process_manager] Process_manager with id 5 has produced 2 elements.
[OK][factory_manager] Process_manager with id 5 has finished.
[OK][factory_manager] Finishing.
```

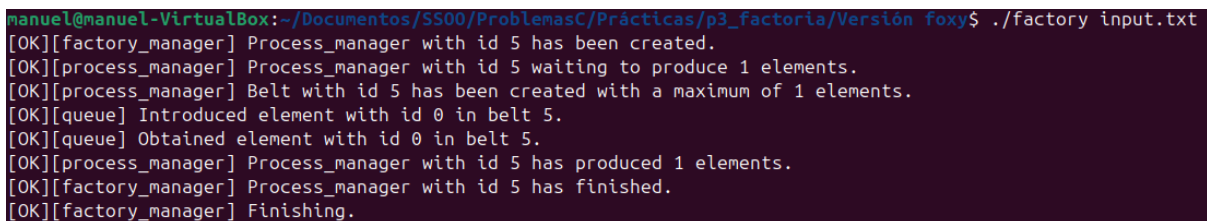
**Test 10:** Vamos a probar un caso válido con el mínimo de elementos por cinta siendo la cinta a su vez de tamaño mínimo (1 5 1 1)

- Fichero de entrada:



```
input.txt x
Prácticas > p3_factoria > Versión foxy > input.txt
1 1 5 1 1
```

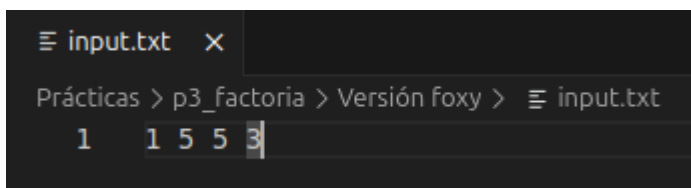
- Salida:



```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 5 has been created.
[OK][process_manager] Process_manager with id 5 waiting to produce 1 elements.
[OK][process_manager] Belt with id 5 has been created with a maximum of 1 elements.
[OK][queue] Introduced element with id 0 in belt 5.
[OK][queue] Obtained element with id 0 in belt 5.
[OK][process_manager] Process_manager with id 5 has produced 1 elements.
[OK][factory_manager] Process_manager with id 5 has finished.
[OK][factory_manager] Finishing.
```

**Test 11:** Vamos a probar un caso válido sin llegar a llenar la cinta (1 5 5 3)

- Fichero de entrada:



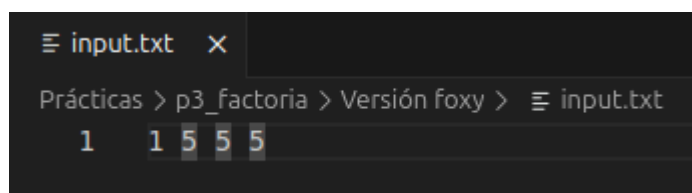
```
input.txt x
Prácticas > p3_factoria > Versión foxy > input.txt
1 1 5 5 3
```

- Salida:

```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 5 has been created.
[OK][process_manager] Process_manager with id 5 waiting to produce 3 elements.
[OK][process_manager] Belt with id 5 has been created with a maximum of 5 elements.
[OK][queue] Introduced element with id 0 in belt 5.
[OK][queue] Introduced element with id 1 in belt 5.
[OK][queue] Introduced element with id 2 in belt 5.
[OK][queue] Obtained element with id 0 in belt 5.
[OK][queue] Obtained element with id 1 in belt 5.
[OK][queue] Obtained element with id 2 in belt 5.
[OK][process_manager] Process_manager with id 5 has produced 3 elements.
[OK][factory_manager] Process_manager with id 5 has finished.
[OK][factory_manager] Finishing.
```

**Test 12:** Vamos a probar un caso válido llenando la cinta (1 5 5 5)

- Fichero de entrada:



The screenshot shows a text editor window titled 'input.txt'. The content of the file is '1 1 5 5 5'. The editor's breadcrumb path is 'Prácticas > p3\_factoria > Versión foxy > input.txt'.

- Salida:

```
manuel@manuel-VirtualBox:~/Documentos/SS00/ProblemasC/Prácticas/p3_factoria/Versión foxy$ ./factory input.txt
[OK][factory_manager] Process_manager with id 5 has been created.
[OK][process_manager] Process_manager with id 5 waiting to produce 5 elements.
[OK][process_manager] Belt with id 5 has been created with a maximum of 5 elements.
[OK][queue] Introduced element with id 0 in belt 5.
[OK][queue] Introduced element with id 1 in belt 5.
[OK][queue] Introduced element with id 2 in belt 5.
[OK][queue] Introduced element with id 3 in belt 5.
[OK][queue] Introduced element with id 4 in belt 5.
[OK][queue] Obtained element with id 0 in belt 5.
[OK][queue] Obtained element with id 1 in belt 5.
[OK][queue] Obtained element with id 2 in belt 5.
[OK][queue] Obtained element with id 3 in belt 5.
[OK][queue] Obtained element with id 4 in belt 5.
[OK][process_manager] Process_manager with id 5 has produced 5 elements.
[OK][factory_manager] Process_manager with id 5 has finished.
[OK][factory_manager] Finishing.
```

### 3. Conclusión

La realización de esta práctica nos ha ayudado a entender el funcionamiento de los semáforos y de los mutex en Unix. Aunque solamente hayamos utilizado los mutex y sus condiciones, nos hemos visto obligados a estudiar el funcionamiento también de los semáforos para poder comprender cómo sincronizar correctamente todos los hilos. Lo que más claro nos ha quedado ha sido el problema del productor consumidor que hay dentro de la práctica con la escritura y lectura sobre las colas cíclicas.

Nos hemos visto en situaciones bastante complicadas sobre todo a la hora de sincronizar la lectura y escritura ya que nos ocurrían una serie de condiciones de carrera que no éramos capaces de localizar.

En conclusión nos ha parecido una muy buena práctica para poder estudiar a fondo la sincronización de hilos y procesos en Unix.