Empaquetador de tornillos

Ejercicio N° 2

Objetivos	● Diseño y construcción de sistemas orientados a objetos ● Diseño y construcción de sistemas con procesamiento concurrente ● Encapsulación de Threads en clases ● Protección de los recursos compartidos
Instancias de Entrega	**Entrega 1:** clase 6 (17/04/2018). **Entrega 2:** clase 8 (01/05/2018).
Temas de Repaso	● Threads en C++11 ● Clases en C++11
Criterios de Evaluación	● Criterios de ejercicios anteriores ● Orientación a objetos del sistema ● Empleo de estructuras comunes C++ (string, fstreams, etc) en reemplazo de su contrapartida en C (char*, FILE*, etc) ● Uso de **const** en la definición de métodos y parámetros ● Empleo de constructores y destructores de forma simétrica ● Buen uso del stack para construcción de objetos automáticos ● Ausencia de condiciones de carrera e interbloqueo en el acceso a recursos ● Buen uso de Mutex y Monitores para el acceso a recursos compartidos

Índice

Introducción

A fin de mejorar su rendimiento, una planta de producción de tornillos desea automatizar el empaquetado de sus productos. Para esto, se solicitó una aplicación que pueda coordinar la salida de los actuales clasificadores de tornillos.

Descripción

La fábrica ScrewU produce tornillos de distintos tipos, materiales y dimensiones. Los tornillos producidos pasan por un control de calidad y luego son clasificados por un sistema electrónico basado en PLCs. Estos clasificadores se conectan a un sistema de archivos en red, de forma que cada clasificador es visto como un archivo de nombre variable.
El empaquetador lee estos archivos para conocer el estado de los clasificadores y lo que ellos entregan. Un clasificador termina de operar cuando se llega al final del archivo.
Como las actualizaciones de los dispositivos son asincrónicas, se requiere que cada clasificador sea atendido en un hilo dedicado en la aplicación del empaquetador.

Configuración del empaquetador

El empaquetador recibe la configuración de los tornillos mediante un archivo de texto plano con el siguiente formato:

```
<id1>=<nombre1>,<limite1>
...
<idN>=<nombreN>,<limiteN>
```

Donde *id* será el identificador del tipo de tornillo, *nombre* un nombre descriptivo, y *límite* la cantidad de tornillos necesaria para armar un paquete. Un paquete puede contener tornillos de distinto ancho, pero de un único tipo.

Formato de los archivos de clasificadores

Los clasificadores son archivos binarios que comienzan con el nombre del clasificador, delimitado por un byte 0, seguido de N bloques de 4 bytes que representan las salidas de los clasificadores.

Estos bloques de 4 bytes, en *big endian*, se interpretan de la siguiente forma:

- Los 5 bits más significativos corresponden al tipo de tornillo.
- Los 22 bits siguientes indican la cantidad de tornillos clasificados en ese momento.
- Los últimos 5 bits representan el ancho los tornillos en mm.

```
<Nombre>\0<tuplas de 4 bytes>
```

Si los 4 bytes representan el valor 0xFF FF FF FF, el clasificador se considera atascado, notificándose en la salida de error estándar.

Formato de línea de comandos

La línea de comandos para ejecutar la aplicación es

```
./tp <config> <operaciones1.bin> [<operacionesN.bin> ...]
```

Entrada y salida estándar

La aplicación no recibe datos mediante la entrada estándar.
La **salida estándar** será utilizada para informar los siguientes eventos:

- Por cada archivo que se abre, se imprime una línea con el siguiente formato

```
<filename>: se establece conexion con el dispositivo <nombre>
```

El orden de las líneas será dado por el orden en que aparecen los nombres de archivo en el vector de argumentos.

- Si un paquete se llena, se imprimirá una línea con el siguiente formato donde la mediana es la mediana de los anchos de los tornillos en mm:

```
Paquete listo: <n> tornillos de tipo <nombre> (mediana: <mediana>)
```

- Cuando todos los clasificadores terminan de operar, el empaquetador informa sobre los tornillos remanentes:

```
# Informe de remanentes
* <n1> tornillos de tipo <nombre1>
* <n2> tornillos de tipo <nombre2>
[...]
```

(En el informe **los tornillos se ordenan por el número de tipo**)

La **salida de error estándar** será utilizada para informar los siguientes eventos:

Por cada archivo que no pudo ser abierto, se imprime el siguiente mensaje:

```
<filename>: no se pudo conectar con el dispositivo
```

- Si se ingresa un tipo inválido de tornillo, se imprime el siguiente mensaje:

```
Tipo de tornillo invalido: <n>
```

Donde es *n* el id inválido.

- Si el clasificador se atasca, imprime el siguiente mensaje:

```
<Nombre clasificador> atascado
```

Donde es el nombre leído en el archivo del clasificador.

Ejemplos de ejecución

Clasificación simple

Archivo config.cfg

```
1=Cabeza plana,100
2=Cabeza puntiaguda,80
3=Cabeza redondeada,80
```

Archivo operador1.bin

Realizamos un hexdump del archivo de entrada

```
00000000  53 49 4d 50 4c 45 34 32  00 08 00 01 5a 08 00 02  |SIMPLE42....Z...|
00000010  9c 08 00 05 19 10 00 03  ca 08 00 06 5d           |............]|
```

`53 49 4d 50 4c 45 34 32 00` Es la representación en hexadecimal de “SIMPLE42” con el delimitador 0

Luego se marca en negrita el primer byte cada una de las operaciones. Nótese como todas son de 4 bytes.

La primera operación es la siguiente. Para facilitar la explicación se muestran los 4 bytes en hexadecimal (izquierda) y en binario (derecha):

08 00 01 5a -> 0000 1000 0000 0000 0000 0001 0101 1010
0000 1 -> Tornillo tipo 1
000 0000 0000 0000 0001 010 -> 10 tornillos
1 1010 -> Ancho 26

Notese como los 4 bytes son interpretados como 3 campos de 5, 22 y 5 bits cada uno.

Las siguientes operaciones son:

08 00 02 9c -> 0000 1000 0000 0000 0000 0010 1001 1100
0000 1 -> Tornillo tipo 1
000 0000 0000 0000 0010 100 -> 20 tornillos
1 1100 -> Ancho 28

08 00 05 19 -> 0000 1000 0000 0000 0000 0101 0001 1001
0000 1 -> Tornillo tipo 1
000 0000 0000 0000 0101 000 -> 40 tornillos
1 1001 -> Ancho 25

10 00 03 ca -> 0001 0000 0000 0000 0000 0011 1100 1010
0001 0 -> Tornillo tipo 2
000 0000 0000 0000 0011 110 -> 30 tornillos
1 1010 -> Ancho 26

08 00 06 5d -> 0000 1000 0000 0000 0000 0110 0101 1101

```
0000 1 -> Tornillo tipo 1
000 0000 0000 0000 0110 010 -> 50 tornillos
1 1101 -> Ancho 29
```

Salida esperada

La salida esperada para esta ejecución es

```
operador1.bin: se establece conexion con el dispositivo SIMPLE42
Paquete listo: 100 tornillos de tipo Cabeza plana (mediana: 27)
# Informe de remanentes
* 20 tornillos de tipo Cabeza plana
* 30 tornillos de tipo Cabeza puntiaguda
* 0 tornillos de tipo Cabeza redondeada
```

Notar que la mediana se calcula como el valor central en un arreglo de largo N donde N es la cantidad de tornillos del paquete. Si el arreglo es de longitud par, la mediana se define como el promedio entre los 2 valores centrales (N/2 - 1 y N/2)[1].

Clasificación doble entrada

Archivo config.cfg

```
1=Cabeza plana,100
2=Cabeza puntiaguda,80
3=Cabeza redondeada,80
```

Archivo doble-a.bin

```
00000000  44 4f 42 4c 45 2d 41 00  10 00 02 9a               |DOBLE-A.....|
```

10 00 02 9a -> 0001 0000 0000 0000 0000 0010 1001 1010
0001 0 -> Tornillo tipo 2
000 0000 0000 0000 0010 100 -> 20 tornillos
1 1010 -> Ancho 26

Archivo doble-b.bin

```
00000000  44 4f 42 4c 45 2d 42 00  10 00 0b 5b               |DOBLE-B....[|
```

10 00 0b 5b -> 0001 0000 0000 0000 0000 1011 0101 1011
0001 0 -> Tornillo tipo 2
000 0000 0000 0000 1011 010 -> 90 tornillos
1 1011 -> Ancho 27

Salida esperada

Si ejecutamos

```
./tp doble-a.bin doble-b.bin
```

La salida esperada para esta ejecución es

```
doble-a.bin: se establece conexion con el dispositivo DOBLE-A
doble-b.bin: se establece conexion con el dispositivo DOBLE-B
Paquete listo: 80 tornillos de tipo Cabeza puntiaguda (mediana: 27)
# Informe de remanentes
* 0 tornillos de tipo Cabeza plana
* 30 tornillos de tipo Cabeza puntiaguda
* 0 tornillos de tipo Cabeza redondeada
```

Notar que independiente del orden de ejecución de los clasificadores, la mediana va a ser 27 (pudo haberse guardado primero 20 tornillos de ancho 26 y luego 60 de ancho 27, u 80 de ancho 27 y luego los 10 remanentes de ancho 27 más los 20 de ancho 26.

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema debe desarrollarse en ISO C++11.
2. Está prohibido el uso de variables globales.
3. Se deben cumplir las restricciones sobre tamaños especificadas en la sección **Descripción.**

Referencias

[1] Mediana: https://es.wikipedia.org/wiki/Mediana_(estad%C3%ADstica)