UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TECNICA SUPERIOR DE INGENIERIA INFORMATICA

Proyecto de Fin de Carrera de Ingeniero Informático

**Analyzer & Profiler for COBOL Batch**

FRANCISCO JAVIER GONZÁLEZ GRÁNDEZ

Dirigido por: NOMBRE DEL DIRECTOR

Supervisado por: JAVIER VÉLEZ

Curso: CURSO (convocatoria de defensa)





**Analyzer & Profiler for COBOL Batch**

Proyecto de Fin de Carrera de modalidad *oferta específica*

Realizado por: FRANCISCO JAVIER GONZÁLEZ GRÁNDEZ (firma)

Dirigido por: NOMBRE DEL DIRECTOR (firma)

Supervisado por: JAVIER VÉLEZ (firma)

Tribunal calificador:

Presidente: D./Dª

(firma)

Secretario: D./Dª

(firma)

Vocal: D./Dª

(firma)

Fecha de lectura y defensa:

Calificación:

**ÍNDICE**

Siglas, abreviaturas y acrónimos 8

1 Contexto 10

2 Planteamiento de la solución 12

2.1 Análisis de la calidad del código 12

2.2 Análisis del comportamiento 13

2.3 Ámbitos de aplicación 14

2.4 Análisis económico 15

3 Planteamiento de la solución 16

4 Conceptos 17

4.1 COBOL 17

4.2 Estructura 17

4.3 Variables 18

4.4 Flujo de ejecución 20

4.5 Rutinas 21

4.6 Complejidad ciclomática 23

4.7 Consumos 25

5 Bloques 26

5.1 Insercion de variables 26

6 Modelo de datos 28

6.1 Tablas generales 29

6.2 Tablas de aplicación 30

6.3 Tablas de módulo 30

6.4 Tablas de ejecución 31

6.5 Tablas de sesión 31

6.6 Tablas sumarizadas 31

7 Collector 32

7.1 Tipos de mensajes 32

7.1.1 Módulo 33

7.1.2 Párrafo 34

7.1.3 CALL 35

7.1.4 Cobertura 36

8 Datos tecnicos 39

8.1.1 Building DLLs 40

9 Manual de instalación y configuración 42

9.1 Arquitectura hardware 42

9.2 Arquitectura Software 43

9.3 Servidor 44

9.3.1 Sistema de mensajería WebSphere MQ Series 45

9.4 Configuración 45

9.4.1 Servidor 45

9.4.2 MQSeries 46

9.4.3 The Number Of Maximum Files Was Reached, How Do I Fix This Problem? 49

9.5 User Level FD Limits 50

10 Manual de usuario 51

10.1 Parámetros y configuración 51

10.1.1 Fichero de configuración 51

10.1.2 Variables de entorno 52

10.2 SDPParser 53

11 Anexos 57

11.1 Configuración MQ 57

Lista de referencias y bibliografía 58

Anexos 61

A1. Definición del modelo de datos 62

A2. Definición del sistema de mensajería 74

A3. Script de compilación 75

A4. Programas de prueba 78

A5. Software de terceras partes 79

**Índice de figuras**

Diagrama 1: Árbol llamadas estáticas 23

Diagrama 2: Relación entre grupos de tablas 29

Código 1: DOCB0001 - Declaración de variables 19

Código 2: DOCB0101 21

Código 3: DOCB0102 21

Código 3: DOCB0201 21

Siglas, abreviaturas y acrónimos

|  |  |
| --- | --- |
| Concepto | Definición |
|  |  |
| API | *Application Program Interface*  Interfaz de definición de la funciones disponibles por un componente software |
|  |  |
| JMS | *Java Message Service*  Especificación del sistema de mensajería basado en Java |
|  |  |
| MIPS | Millones de Instrucciones por Segundo. En plataformas IBM es una unidad de medida del consumo de CPU |
|  |  |
| *Porting* | Acción de tomar un código fuente y convertirlo en otro, bien sea en el mismo lenguaje pero capaz de ejecutarse en una máquina diferente o bien convirtiéndolo a otro lenguaje de programación |
|  |  |
| Q&A | *Quality & Assurance*  En este documento hace referencia al departamento responsable de establecer y mantener los procedimientos y controles de calidad del software |
|  |  |
| SGDB | Sistema Gestor de Bases de Datos. |
|  |  |
| TIC | Tecnologías de la información y las comunicaciones. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Contexto

Actualmente, cuando se habla de Sistemas de Información y de lenguajes de programación, resulta normal pensar en lenguajes de programación orientados a objetos como Java, interpretados como Javascript, PHP, Perl, y a veces, pero no con tanta frecuencia, C y C++, como indica, por ejemplo, el informe Introducing the Gartner Programming Language Index for 2014 [GAR1] o el Indice TIOBE [TIO1].

Sin embargo, tal vez no somos conscientes de que según algunos estudios realizados a lo largo del año 2.013 por empresas como por ejemplo, MicroFocus, Gartner, IT Media o ComputerPro [CP1] entre otros, reflejan otro tipo de información, como por ejemplo:

* Durante el año 2013 se crearon más programas COBOL que páginas Web
* COBOL soporta cada día el 90% de los sistemas de negocio de Fortune 500
* COBOL soporta el 70% de la lógica de negocio crítica
* COBOL procesa el 85% de las transacciones comerciales diarias
* Un ciudadano interactúa una media de 10 veces al día con un programa COBOL

Lo cual da una idea de la relevancia que todavía mantiene este lenguaje de programación.

Por otro lado, y a diferencia de otros tipos de aplicaciones desarrolladas en otros lenguajes, suele ser bastante habitual que los sistemas hardware/software sobre los que se ejecutan estas aplicaciones tengan un coste de explotación variable en función del uso de los recursos consumidos o necesitados (MIPS, Memoria, Procesadores, etc.) para su ejecución, por lo que el coste asociado a la generación de un programa COBOL no se limita, simplificando, al coste de su codificación, pruebas y mantenimiento, si no que se extiende a lo largo de todo su ciclo de vida y supone un coste para la empresa propietaria del programa cada vez que este es ejecutado.

Por lo tanto, los conceptos de corrección y eficiencia de un programa asociados al desarrollo de aplicaciones, y obviados tan a menudo, en aras de la rapidez, economía y reducción de los plazos de entrega, cobran en este entorno una importancia mucho mayor de la que ya de por sí debería tener para cualquier sistema empresarial ya que cada vez que se ejecuta un programa COBOL no óptimo, este repercute directamente en la cuenta de resultados de la empresa.

A pesar de todo ello, y a diferencia de otros lenguajes como C o Java, prácticamente no existen herramientas que permitan realizar un control de calidad del código desarrollado antes de ser ejecutado, y los que hay son bastantes limitados y con un fuerte componente manual.

Por mencionar solo un ejemplo, sin embargo muy habitual en este entorno, el método de validación de la corrección de un programa en una de las primeras entidades financieras del mundo, se basa en entregar a un departamento de control de calidad un listado en papel con los resultados de las pruebas realizadas.

Esta situación se vuelve más crítica cuando se trata de evaluar el rendimiento y el comportamiento de los programas en entornos productivos, puesto que los administradores de los sistemas no tienen la información necesaria para evaluar si un programa se está comportando adecuadamente y las herramientas disponibles no están orientadas al análisis de programas en concreto sino más bien al comportamiento del sistema y no están disponibles ni para los equipos de desarrollo ni para los departamentos responsables

# Planteamiento de la solución

Este proyecto presenta una solución a los problemas descritos en el apartado anterior utilizando como dialecto del lenguaje la implementación desarrollada por el proyecto open-source GNUCobol [GNUC] abordando las dos perspectivas:

1. Análisis de la calidad del código fuente desarrollado
2. Análisis del comportamiento del programa en un entorno productivo

## Análisis de la calidad del código

El enfoque utilizado para realizar este proceso se basa en “pseudo-compilar” el código fuente utilizando las herramientas de análisis léxico, sintáctico y semántico típicas en el desarrollo de compiladores para construir los arboles semánticos correspondientes, los grafos de ejecución y extraer la información relevante que permita analizar la complejidad del código desarrollado, su adecuación a las directivas de desarrollo definidas por la entidad para la cual se ha desarrollado y su adecuación a unas métricas de calidad del software homogéneas.

Para clarificar este proceso XXXXX

Sentencias no permitidas

El ejemplo más evidente sería la sentencia GO TO, que, como la propia sentencia indica, realiza un salto incondicional a otra parte del código. Esta sentencia se sigue manteniendo en todas las versiones del lenguaje, pero está prohibido su uso en todas las instalaciones de Sistemas de Información.

Al ejecutar el “pseudo-compilador” sobre un código que tuviera esta sentencia, y de acuerdo con las normas definidas en la instalación, el sistema generará un error indicando que existen instrucciones no permitidas, impidiendo que el programa se compile correctamente.

Análisis de la legibilidad

El sistema puede avisar de situaciones, que aunque correctas desde el punto de corrección del programa, no se consideran adecuadas:

|  |  |
| --- | --- |
| MOVE  UNA-VARIABLE  TO  OTRA-VARIABLE | MOVE UNA-VARIABLE TO OTRA-VARIABLE |
|  |  |
| IF A THEN  MOVE A TO B  IF B THEN  MOVE B TO C  IF C THEN  MOVE C TO D. | IF A THEN  MOVE A TO B  IF B THEN  MOVE B TO C  IF C THEN  MOVE C TO D. |

Análisis de la complejidad

El sistema calcula la complejidad ciclomática de sus componentes con lo que, se puede evitar que existan bloques de código demasiado complejos que deberían ser refactorizados en bloques más sencillos y permite establecer el número de casos de prueba necesarios para garantizar la corrección del programa.

## Análisis del comportamiento

Para poder realizar el análisis del comportamiento real del programa en un entorno productivo el enfoque utilizado es, basándonos en el mismo proceso de pseudo-compilación, inyectar código especial en los puntos críticos del código fuente que permita monitorizar en tiempo de ejecución el flujo real del programa y los consumos del mismo, a este código especial lo denominaremos TRAP.

Es decir, cuando se procesa un código fuente con el sistema el resultado es **otro código fuente** funcionalmente equivalente al original pero que va aportando información acerca de las partes de código ejecutadas realmente y de sus consumos.

En el momento de diseñar este componente resulta crítico considerar que:

* La inclusión de Traps en el código hace que este tenga que ejecutar más código, lo cual, inevitablemente produce un consumo más de máquina y un tiempo de ejecución superior.
* Como ya se ha indicado en el apartado anterior, la ejecución de un programa tiene un coste monetario dependiente del consumo de máquina realizado,

Para minimizar el impacto en estos dos aspectos:

* El componente de captura de la información se ha desarrollado en C, como lenguaje más óptimo desde el punto de vista de la máquina después del ensamblador, y se ha minimizado su proceso de forma que únicamente sea responsable de capturar la información relevante.
* La información recolectada es enviada para su tratamiento y análisis de la información a un sistema externo, no productivo y cuyos costes no dependan del uso de que se haga del mismo.

## Ámbitos de aplicación

Dado que es un sistema que realiza análisis de programas COBOL, su ámbito de aplicación es, lógicamente, cualquier instalación que tenga software desarrollado en ese lenguaje. Esto supone, simplemente en España, entre otros:

* Todas las entidades financieras de Banca Retail.
* Todas las compañías de telecomunicaciones
* La mayoría de las empresas energéticas
* Transportes: Iberia, Renfe, Adif
* Grandes empresas de servicios y distribución

Dentro de estas empresas determinadas áreas o departamentos que requieran de herramientas de control del software, como por ejemplo:

* Departamento de Q&A: Como herramienta central para establecer los criterios y controles de calidad del software desarrollado.
* Departamentos de operaciones y explotación: Como herramienta para analizar problemas en tiempo de ejecución.
* Departamentos de desarrollo: Como herramienta de control de calidad y de perfilado de programas
* Departamentos operativos que soporten en su contabilidad interna los costes de ejecución de su software o que necesiten herramientas de análisis para prever los recursos necesarios en TIC a medio plazo

Por otro lado, y basándonos en la tecnología de inyección de código, la herramienta se puede utilizar como base para:

* Proyectos de porting de software
* Proyectos de optimización de código
* Proyectos de análisis de calidad del software
* En general, cualquier proyecto que requiera un análisis sintáctico o semántico de código fuente

## Análisis económico

# Conceptos

## COBOL

XXXXXXXXXXXXXXX

## Estructura

Un programa fuente COBOL está estructurado en divisiones que agrupan un conjunto de instrucciones del lenguaje asociadas a un propósito común. No todas las divisiones son obligatorias pero si existen deben indicarse de acuerdo con el siguiente orden:

IDENTIFICATION DIVISION.

PROGRAM-ID. Nombre-del-programa.

otras-sentencias-de-identification

[ENVIRONMENT DIVISION.

contenido-environment-division]

[DATA DIVISION.

contenido-data-division]

PROCEDURE DIVISION [USING …].

codigo-ejecutable

[END-PROGRAM]

Código 1: Estructura COBOL

* IDENTIFICATION DIVISION

Contiene información básica relativa al programa, la única clausula obligatoria es PROGRAM-ID que identifica el nombre del programa que será el utilizado para su invocación.

Se considera una buena práctica que este nombre coincida con el nombre del archivo fuente del mismo; sin embargo, en determinados sistemas operativos, como por ejemplo zOS de IBM o VOS de Stratus, se limita el nombre de los archivos a 8 caracteres

* ENVIRONMENT DIVISION

Contiene la información relativa al entorno en el cual el programa será compilado y ejecutado: memoria, tipo de ordenador, repositorio de funciones externas, etc. En esta división se deben declarar la lista y características de todos los ficheros a los que el programa puede acceder.

* DATA DIVISION

Contiene la declaración de todas las variables que pueden ser utilizadas por el programa.

* PROCEDURE DIVISION

Contiene el código del programa.

## Variables

En un programa COBOL todas las variables deben haber sido declaradas en DATA DIVISION y son accesibles a todo el programa.

Una de las principales características que diferencia a este lenguaje de otros lenguajes de programación estructurada es que las variables se agrupan en números de nivel formando estructuras de tipo registro que, a su vez, pueden ser “*redefinidas*” formando estructuras unión.

Desde el punto de vista de los números de nivel, se aplican las siguientes restricciones:

* Nivel 77: Define una variable que está dividida en otras variables.
* Nivel 01: Define una variable o un conjunto de variables.
* Niveles 02-49: Definen variables o conjuntos de ellas que están incluidas en un nivel superior, siendo el primer nivel obligatoriamente el nivel 01.

Cuando se referencia una variable, el valor de esta comprende todas las variables que dependen de ella. Por ejemplo, la ejecución del siguiente programa:

\* SDP DESCRIPTION

\* DOC - Demuestra el uso de los niveles en la definicion de

\*> las variables

\* SDP END

IDENTIFICATION DIVISION.

PROGRAM-ID. DOCB0001.

DATA DIVISION.

WORKING-STORAGE SECTION.

77 VAR77 PIC X(10) VALUE 'VAR77'.

01 VAR01.

03 ABC PIC X(03) VALUE 'ABC'.

03 NIVEL03.

05 LETRAD PIC X(01) VALUE 'D'.

05 LETRAE PIC X(01) VALUE 'E'.

05 LETRAF PIC X(01) VALUE 'F'.

PROCEDURE DIVISION.

PERFORM DISPLAY-VARIABLES.

MOVE SPACES TO NIVEL03.

DISPLAY 'TRAS INICIAR NIVEL03 A ESPACIOS'.

PERFORM DISPLAY-VARIABLES.

STOP RUN.

DISPLAY-VARIABLES.

DISPLAY 'VARIABLE VAR77 ES : ' VAR77.

DISPLAY 'VARIABLE VAR01 ES : ' VAR01.

DISPLAY 'VARIABLE NIVEL03 ES : ' NIVEL03.

DISPLAY 'VARIABLE LETRAE ES : ' LETRAE.

Código 1: DOCB0001 - Declaración de variables

Produce el siguiente resultado:

$ ./DOCB0001.exe

VARIABLE VAR77 ES : VAR77

VARIABLE VAR01 ES : ABCDEF

VARIABLE NIVEL03 ES : DEF

VARIABLE LETRAE ES : E

TRAS INICIAR NIVEL03 A ESPACIOS

VARIABLE VAR77 ES : VAR77

VARIABLE VAR01 ES : ABC

VARIABLE NIVEL03 ES :

VARIABLE LETRAE ES :

## Flujo de ejecución

Salvo que se utilice la cláusula ENTRY el flujo del programa empieza en la primera sentencia que aparezca después de PROCEDURE DIVISION y es ejecutado secuencialmente hasta que se produce una de las siguientes situaciones:

1. Encuentra el final de fichero
2. Encuentra END PROGRAM, indicando de manera explícita el fin del fichero fuente
3. Encuentra STOP RUN, indicando el final de la ejecución del proceso
4. Encuentra GOBACK, indicando el final de la ejecución del módulo actual

Esto puede dar lugar a comportamientos que los programadores en otros lenguajes de programación puedan considerar “extraños”:

Por ejemplo, el siguiente código produce el mismo resultado:

|  |  |  |
| --- | --- | --- |
| PROCEDURE DIVISION.  DISPLAY 'Hello World'. |  | PROCEDURE DIVISION.  INICIO.  DISPLAY 'Hello World'. |
|  |  |  |
| **$ ./DOCB0101.exe**  **Hello World** |  | **$ ./DOCB0102.exe**  **Hello World** |
| Código 2: DOCB0101 |  | Código 3: DOCB0102 |

Pero el siguiente código genera un resultado diferente del esperado:

|  |
| --- |
| PROCEDURE DIVISION.  PERFORM INICIO.  INICIO.  DISPLAY 'Hello World'. |
|  |
| **$ ./DOCB0201.exe**  **Hello World**  **Hello World** |
| Código 4: DOCB0201 |

## Rutinas

El lenguaje COBOL está fuertemente orientado a la modularización y la reutilización del código en base al uso de rutinas o módulos.

Una rutina es un programa COBOL normal que es invocado por otro programa y puede o no recibir parámetros por referencia (opción por defecto) o por valor.

Existen dos modos de invocación a una rutina:

* Llamada estática: CALL “nombre\_rutina” [USING parámetros]
* Llamada dinámica: CALL variable [USING parámetros]

El uso de uno u otro modo afecta directamente al modo de funcionamiento del programa, cuando se utilizan llamadas estáticas a módulos:

* En el momento de compilación se deben indicar todos los módulos utilizados
* Se genera un único ejecutable con todos los módulos incluidos en el mismo. Por esto es necesario declararlos todos en el momento de compilación, ya que en caso contrario, se produce un error en tiempo de ejecución
* Cualquier cambio posterior en las rutinas incluidas no se ven reflejadas en el programa principal.

Cuando se utiliza una llamada dinámica:

* La rutina no es incluida en el ejecutable y se carga en tiempo de ejecución como si fuera una librería dinámica
* La rutina es compilada de manera independiente por lo que siempre se utiliza la última versión de la misma

En cualquiera de los dos modos de invocación el módulo invocado es cargado en el espacio de direcciones del módulo principal **una única vez**; es decir, si en alguno de los módulos implicados en una unidad de ejecución ya está cargada una determinada rutina, esta no se vuelve a cargar y su estado interno se mantiene a lo largo de toda la ejecución.

Para aclarar este concepto consideremos el siguiente árbol de módulos, donde las flechas continuas indican llamadas estáticas, las discontinuas llamadas dinámicas, y los números el orden de ejecución:



Diagrama 1: Árbol llamadas estáticas

Cada programa, cuando es invocado, incrementa en uno un contador y muestra su valor. El resultado es:

$ ./DOCB0301.exe

EJECUTA DOCB0311 DESDE DOCB0301 CON CONTADOR = 001

EJECUTA DOCB0321 DESDE DOCB0311 CON CONTADOR = 001

EJECUTA DOCB0322 DESDE DOCB0311 CON CONTADOR = 001

EJECUTA DOCB0321 DESDE DOCB0322 CON CONTADOR = 002

EJECUTA DOCB0321 DESDE DOCB0301 CON CONTADOR = 003

## Complejidad ciclomática

La complejidad ciclomática según fue definida por Thomas McCabe en su célebre articulo A Complexity Measure [CAB1] establece un método para medir y controlar el número de caminos diferentes que puede seguir el flujo de un programa, y en consecuencia, identifica el conjunto de pruebas mínimo que sería necesario realizar para garantizar que se han ejecutado al menos una vez todas las sentencias de código de ese programa.

Considerando que un programa ejecuta sus instrucciones secuencialmente, la única forma de que se establezca un cambio en el flujo del programa es a través de las denominadas “sentencias de control de flujo”. En COBOL las únicas sentencias de control de flujo disponibles son:

* Sentencia IF: Ejecuta el conjunto de sentencias si se cumple la condición
* Sentencia EVALUATE: Es una forma compacta de agrupar un conjunto de sentencias IF dependientes de una variable
* Clausula UNTIL en sentencias PERFORM: La sentencia bifurca el control del programa mientras se cumplan las condiciones establecidas en la clausula UNTIL

Cada una de ­⌐estas situaciones se pueden modelar como un grafo de entre los siguientes:







## Consumos

# Bloques

Each DIVISION may consist of a variety of SECTIONs and each SECTION consists of one or more PARAGRAPHs. A PARARAPH consists of SENTENCEs, each of which consists of one or more STATEMENTs.

Desde el punto de vista del código, este se puede ver formado por:

Secciones : Una sección esta formada por uo o mas párrafos

Parrafo: Un párrafo esta formado uno o mas sentencias

Sentencia: UN conjunto de bloques

Bloque: Un conjunto de instrucciones que se ejecutarán como una unidad; en el sentido de que, si el flujo de programa alcanza ese punto, se ejecutarán todas ellas

Instrucción: Cada uno de los verbos del lenguaje

## Insercion de variables

Las variables necesarias para el fucnionamiento se alamacenan en Working

Puede no haber esa sección, entonces se mira si hay data división, si la hay hay otras cosas para ponerla en su sitio

La secuencia es local-storage, linkage, report, screen si no hay ninguna de ellas

Va encima de procedure

Si no hay data división

Va antes de procedure incluyendo data division

# Modelo de datos

El modelo de datos está estructurado en función del componente del sistema que lo alimenta y el bloque funcional que representa

|  |  |
| --- | --- |
| **Prefijo** | **Descripción** |
| CFG | Tablas de configuración y mensajes |
| LOG | Tablas de gestión de mensajes de log |
| SDP | Tablas base del sistema |
| SDP | Tabla con la información relativa a las áreas, aplicaciones y módulos registrados en el sistema |
| MOD | Tablas con la información estática de un determinado modulo |
| TRP | Mensajes generados por el componente Trapper durante una sesión |
| SES | Datos de la sesión ejecutada |
| SUM | Datos sumarizados de un determinado módulo |

Las relaciones que se establecen entre los diferentes grupos de tablas se muestran en el siguiente diagrama



Diagrama 2: Relación entre grupos de tablas

## Tablas generales



## Tablas de aplicación

## Tablas de módulo

## Tablas de ejecución

## Tablas de sesión

## Tablas sumarizadas

# Collector

El collector es el responsable de capturar los diferentes eventos que se van produciendo durante la ejecución del programa impactando lo menos posible en el sistema objeto de análisis.

Por ello este componente se implementa mediante una arquitectura Cliente/Servidor cuyo escenario más sencillo se muestra en la siguiente figura:



Ilustración 1: Arquitectura básica Collector

La idea subyacente a esta arquitectura se basa en los siguientes criterios:

## Tipos de mensajes

Los mensajes enviados al servidor son mensajes en modo datagrama separados por el carácter delimitador punto y coma (;), de manera que en función del tipo de mensaje puede contener diferente información.

Los mensajes que se generan son:

* Módulo: Contiene la información relativa al consumo asociado a un determinado módulo.
* Parráfo: contiene la información relativa a los tiempos asociados a un determinado párrafo.
* Llamada: Contiene la informaci

### Módulo

Contiene la información relativa a los consumos realizados dentro de un determinado módulo.

| **Orden** | **Campo** | **Descripción** |
| --- | --- | --- |
| 1 | Tipo | Tipo de mensaje.  Valor: **1** |
| 2 | Id Sesion | Contiene el identificador único de la sesión |
| 3 | Id Modulo | Contiene el identificador único del módulo.  Cuando este valor coincide con el identificador de la sesión los tiempos obtenidos indican los tiempos totales de ejecución del programa |
| 4 | Nombre | Nombre del módulo ejecutado |
| 5 | Elapsed | Tiempo total transcurrido en microsegundos |
| 6 | Kernel | Tiempo total de CPU consumido en modo Kernel |
| 7 | User | Tiempo total de CPU consumido en modo Usuario |
| 8 | Elapsed In | Tiempo en microsegundos consumidos dentro del módulo excluyendo todas las llamadas a módulos externos |
| 9 | Kernel In | Tiempo de CPU consumido en modo Kernel excluyendo el consumido en llamadas externas |
| 10 | User In | Tiempo de CPU consumido en modo Usuario excluyendo el consumido en llamadas externas |

### Párrafo

Contiene la información relativa a los consumos realizados dentro de un determinado módulo.

| **Orden** | **Campo** | **Descripción** |
| --- | --- | --- |
| 1 | Tipo | Tipo de mensaje.  Valor: **2** |
| 2 | Id Sesion | Contiene el identificador único de la sesión |
| 3 | Id Modulo | Contiene el identificador único del módulo. |
| 4 | Nombre | Nombre del párrafo |
| 5 | Elapsed | Tiempo total transcurrido en microsegundos |
| 6 | Kernel | Tiempo total de CPU consumido en modo Kernel |
| 7 | User | Tiempo total de CPU consumido en modo Usuario |
| 8 | Elapsed In | Tiempo en microsegundos consumidos dentro del módulo excluyendo todas las llamadas a otros párrafos o módulos externos |
| 9 | Kernel In | Tiempo de CPU consumido en modo Kernel excluyendo el consumido en llamadas a otros párrafos o módulos externos |
| 10 | User In | Tiempo de CPU consumido en modo Usuario excluyendo el consumido en llamadas a otros párrafos o módulos externos |

### CALL

Contiene la información relativa a los consumos realizados durante la llamada a un módulo externo

| **Orden** | **Campo** | **Descripción** |
| --- | --- | --- |
| 1 | Tipo | Tipo de mensaje.  Valor: **3** |
| 2 | Id Sesion | Contiene el identificador único de la sesión |
| 3 | Id Modulo | Contiene el identificador único del módulo. |
| 4 | Nombre | Nombre del párrafo |
| 5 | Elapsed | Tiempo total transcurrido en microsegundos |
| 6 | Kernel | Tiempo total de CPU consumido en modo Kernel |
| 7 | User | Tiempo total de CPU consumido en modo Usuario |
| 8 | Elapsed In | Tiempo en microsegundos necesario para realizar la carga del módulo |
| 9 | Kernel In | Tiempo de CPU en modo Kernel necesario para realizar la carga y asignación de recursos para el módulo llamado |
| 10 | User In | Tiempo de CPU en modo Usuario necesario para realizar la carga y asignación de recursos para el módulo llamado |

### Cobertura

Contiene la información relativa a los bloques de código que se han ejecutado durante la sesión

| **Orden** | **Campo** | **Descripción** |
| --- | --- | --- |
| 1 | Tipo | Tipo de mensaje.  Valor: **0** |
| 2 | Id Sesion | Contiene el identificador único de la sesión |
| 3 | Id Modulo | Contiene el identificador único del módulo. |
| 4 | Cobertura | Tabla de indicadores con las marcas de ejecución de bloques.  Es una lista de caracteres donde el valor ‘1’ indica que el bloque asociado a su posición relativa dentro de la lista de caracteres se ha ejecutado |

|  |  |
| --- | --- |
|  |  |
| 1 | Tipo de mensaje |
| 2 | Identificador de la sesión |
| 3 | Identificador del modulo |
| 4 | Nombre del componente |
| 5 | Tiempo total consumido en el compon |
|  |  |
|  |  |

|  |
| --- |
| #!/bin/bash |
|  |
| mvn deploy:deploy-file -Durl=sftp://web.sourceforge.net:/home/project-web/emustudio/htdocs/repository -DrepositoryId=emustudio-repository -Dfile=./java-cup-runtime-0.11a.jar -DpomFile=./pom.xml |
|  |

GiTHUB:

https://github.com/

JavierGlez

Fjgg2185

[Fcojavier.glez@gmail.com](mailto:Fcojavier.glez@gmail.com)

<http://www.carlospinan.com/2014/03/11/instalando-opencobol-en-windows-7/>

Oracle

SYS y SYSTEM password jgg

APEX en el Puerto 9090

Acceso SYS/jgg

Sonar

Puerto 9000

Tomcat

Admin/jgg

# Datos tecnicos

Entrada de datos

El formato de un programa COBOL es posicional y distingue tres áreas:

Maquinas:

Serendipity cliente

Usuario: jgonzalez password: jgg la misma que para root

Collector:

SDPxTRPn ->

X Indica el sistema operativo

N indica si es de 32 o 64 bits

### Building DLLs

This page gives only a few simple examples of gcc's DLL-building capabilities. To begin an exploration of the many additional options, see the gcc documentation and website, currently at <http://gcc.gnu.org/>

Let's go through a simple example of how to build a dll. For this example, we'll use a single file myprog.c for the program (myprog.exe) and a single file mydll.c for the contents of the dll (mydll.dll).

Fortunately, with the latest gcc and binutils the process for building a dll is now pretty simple. Say you want to build this minimal function in mydll.c:

#include <stdio.h>

int

hello()

{

printf ("Hello World!\n");

}

First compile mydll.c to object code:

gcc -c mydll.c

Then, tell gcc that it is building a shared library:

gcc -shared -o mydll.dll mydll.o

That's it! To finish up the example, you can now link to the dll with a simple program:

int

main ()

{

hello ();

}

Then link to your dll with a command like:

gcc -o myprog myprog.c -L./ -lmydll

However, if you are building a dll as an export library, you will probably want to use the complete syntax:

gcc -shared -o cyg${module}.dll \

-Wl,--out-implib=lib${module}.dll.a \

-Wl,--export-all-symbols \

-Wl,--enable-auto-import \

# Manual de instalación y configuración

Es esta capítulo se describen los procedimientos necesarios para la instalación y configuración del sistema en base a las siguientes arquitecturas hardware y software

## Arquitectura hardware

Para el servidor del sistema se ha utilizado una distribución de Linux Ubuntu con la siguiente configuración

|  |  |
| --- | --- |
| Diagrama 3: Arquitectura Hardware | **/ - Disco ubuntu.vmdk**  Contiene el sistema con el software necesario para su ejecución.  **/PFC – Disco PFC.vmdk**  Contiene el software, el código y la documentación del proyecto.  En caso de querer ejecutar el sistema en otra plataforma Linux, este disco es el único requerido si la plataforma cumple con los requerimientos de software establecido |
| **/software – Disco software.vmdk**  Contiene el software requerido que no está disponible en los repositorios de software de Ubuntu.  Una vez instalado el software este disco no es requerido y se puede desmontar  **Swap – Disco swap.vmdk**  Dispositivo de paginación. Por motivos de rendimiento es recomendable en un disco separado del sistema operativo. Por otro lado, cuando se utilizan diferentes máquinas virtuales de manera no simultánea, este disco puede ser compartido por todas ellas. | |

## Arquitectura Software

Desde el punto de vista de la arquitectura software, la configuración utilizada se muestra en el siguiente diagrama:



Diagrama 4: Arquitectura Software

|  |  |
| --- | --- |
| Componente | Descripción |
|  |  |
| Sistema Operativo | Distribución Linux Ubuntu Server 15.0.4 para una arquitectura de procesador AMD 64 bits |
|  |  |
| Servidor de base de datos | MySQL Server versión 14.14  La base de datos utilizada se define como **SDP** |
|  |  |
| Sistema de mensajería | Websphere MQ Series 8.0.2 |
|  |  |
| Servidor de páginas | Apache 2 |
|  |  |
| Servidor de aplicaciones | Tomcat Apache 8 |

## Servidor

Para mantener el sistema únicamente con el software necesario se han instalado los siguientes paquetes desde el repositorio central de Ubuntu evitando instalar otros paquetes asociados con el comando:

apt-get install –no-install-recommends *nombre\_del\_paquete*

| Paquete | Contenido |
| --- | --- |
|  |  |
| openSSH | Shell Seguro. Se instala durante la instalación del sistema |
|  |  |
| Vsftpd | Servidor ftp |
|  |  |
| gcc | Compilador GNU C |
|  |  |
| rpm | Gestor de paquetes RPM |
|  |  |
| Make | Utilidad para gestión de compilaciones |
|  |  |
| automake | Herramienta para la generación de Makefiles conformes con los estándares GNU |
|  |  |
| autoconf | Herramienta de generación automática de scripts de configuración |
|  |  |
| default-jre | Open Java Runtime Environment |
|  |  |
| default-jdk | Open Java Development Toolkit |
|  |  |
| open-cobol | Compilador Open Cobol |
|  |  |
| apache2 | Servidor de páginas Apache 2 |
|  |  |
| tomcat8 | Servidor de aplicaciones Apache Tomcat 8 |
|  |  |
| tomcat8-admin | Aplicación de administración Web de Apache Tomcat 8 |
|  |  |
| tomcat8-user | Herramienta para crear instancias de usuarios en Tomcat 8 |
|  |  |
| libapache2-mod-jk | Conector entre Apache y Tomcat |
|  |  |
| mysql-server | Servidor de base de datos MySQL |
|  |  |

Los paquetes que no están disponibles en el repositorio central o que necesitan un tratamiento diferenciado están incluidos en el disco **/software**. Una vez realizada la instalación, este disco podría ser desmontado del sistema.

### Sistema de mensajería WebSphere MQ Series

Los paquetes relativos a MQSeries están empaquetados en formato Debian por lo que pueden mensajes de error en Ubuntu, para evitarlo estos paquetes se deben ejecutar con la orden:

rpm -ivh --nodeps --force-debian paquete\_mq.rpm

| Paquete | Contenido |
| --- | --- |
|  |  |
| mqm/mqlicense.sh | Ejecutar con el parámetro **-accept** para iniciar la instalación |
|  |  |
| mqm/MQSeriesRuntime | Funciones comunes a MQSeries |
|  |  |
| mqm/MQSeriesServer | Servidor MQSeries |
|  |  |
| mqm/MQSeriesClient | Cliente de MQSeries C |
|  |  |
| mqm/MQSeriesJava | API Java y JMS |
|  |  |
| mqm/MQSeriesJRE | Java Runtime |
|  |  |
| mqm/MQSeriesSDK | Librerías y ficheros de cabecera para C |
|  |  |
| mqm/MQSeriesMsg\_es | Mensajes en castellano |
|  |  |

## Configuración

### Servidor

El usuario creado por defecto es **jgonzalez373** con contraseña **uned**

Para poder obtener acceso al sistema como usuario root se recomienda crearle una contraseña mediante el comando:

sudo passwd root

Pedirá la contraseña del usuario jgonzalez373 y la nueva contraseña para el usuario root. Esta contraseña se ha establecido a uned

### MQSeries

En este apartado se indican los pasos a realizar para configurar el sistema de gestión de colas de acuerdo con los requerimientos del sistema.

|  |  |
| --- | --- |
| Objeto | Definición |
| SDP.QMGR | Gestor de colas |
| SDP.INIT | Cola de inicialización |
| SDP.PARSER | Cola para los mensajes generados por SDPParser |
| SDP.TRAPPER | Cola para los mensajes generados por SDPTRAPB |
| SDP.COLLECTOR | Cola de sincronización interna |
| PRC.PARSER | Proceso que se ejecutará cuando existan mensajes en la cola SDP.PARSER |
| PRC.TRAPPER | Proceso que se ejecutará cuando existan mensajes en la cola SDP.TRAPPER |
| PRC.COLLECTOR | Proceso que se ejecutará cuando existan mensajes en la cola SDP.COLLECTOR |

Configurar usuario

Por defecto este paquete se instala en /opt/mqm y crea el usuario mqm y el grupo mqm

Es recomendable establecer una contraseña para ese usuario, en nuestro caso será **uned**, por que el resto del proceso se realizará con ese usuario.

Establecer la instalación principal

MQSeries permite que coexistan varias instalaciones en el mismo servidor pero debe haber una de hecha definida como principal.

Para verificar si se ha creado por defecto ejecutar el comando:

/opt/mqm/bin/dspmqinst

Si no existe una instalación principal, crearla con el comando:

/opt/mqm/bin/setmqinst /opt/mqm

Creación del gestor de colas

Se debe crear un gestor de colas para el sistema, preferentemente como gestor de colas predeterminado ya que en caso contrario será necesario indicarlo de manera explícita en todas las conexiones.

/opt/mqm/bin/crtmqm -q SDP.QMGR

Creación de las colas

Se debe crear un gestor de colas para el sistema, preferentemente como gestor de colas predeterminado ya que en caso contrario será necesario indicarlo de manera explícita en todas

Preparar el sistema para que se ejecute en tiempo de arranque

Una vez definido todo el sistema de mensajería es necesario configurarlo para que se arranque y se detenga de manera sincronizada con el sistema, para ello se provee un script de gestión

#!/bin/sh

### BEGIN INIT INFO

# Provides: MQ Series Service

# Required-Start: $syslog

# Required-Stop: $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: MQ Series services

# Description:

#

### END INIT INFO

############################################################

# Utilidad para arrancar el servidor MQ

#

# Autor : Javier Gonzalez Grandez

# Fecha : Ago - 2015

# Version: 1.0

############################################################

startMQ() {

sudo -u mqm /opt/mqm/bin/strmqm SDP.QMGR

}

stopMQ() {

sudo -u mqm /opt/mqm/bin -i SDP.QMGR

}

if [ $# -eq 0 ] ; then

startMQ

else

if [ "$1" == "start" ] ; then

startMQ

fi

if [ "$1" == "stop" ] ; then

stopMQ

fi

fi

Código 5: Script de gestión de MQSeries

update-rc.d mq defaults 95 02

Rpm

Open-cobol

Make

Automake

autoconf

Vsftpd

Default-jre

Apache2

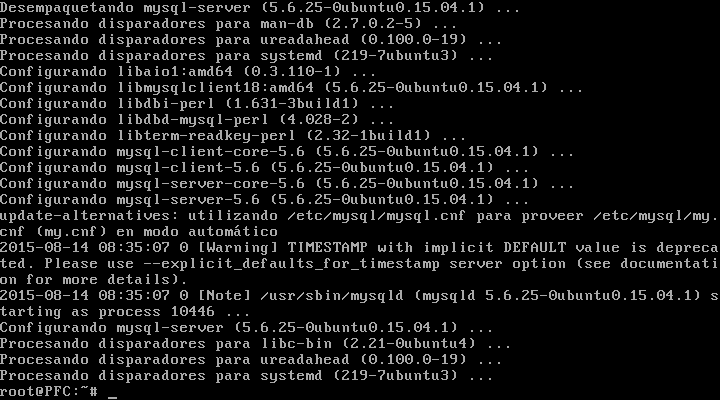
Tomcat8

Tomcat8-admin

Tomcat8-user

Mysql-server (password jgg)

Mirar esto

vs

Configurar vftpd

MQSries

Ajustar ficheros:

http://www.cyberciti.biz/faq/linux-increase-the-maximum-number-of-open-files/

### The Number Of Maximum Files Was Reached, How Do I Fix This Problem?

Many application such as Oracle database or Apache web server needs this range quite higher. So you can increase the maximum number of open files by setting a new value in kernel variable /proc/sys/fs/file-max as follows (login as the root):  
# sysctl -w fs.file-max=100000  
Above command forces the limit to 100000 files. You need to edit /etc/sysctl.conf file and put following line so that after reboot the setting will remain as it is:  
# vi /etc/sysctl.conf  
Append a config directive as follows:  
fs.file-max = 100000  
Save and close the file. Users need to log out and log back in again to changes take effect or just type the following command:  
# sysctl -p  
Verify your settings with command:  
# cat /proc/sys/fs/file-max  
OR  
# sysctl fs.file-max

## User Level FD Limits

The above procedure sets system-wide file descriptors (FD) limits. However, you can limit httpd (or any other users) user to specific limits by editing /etc/security/limits.conf file, enter:  
# vi /etc/security/limits.conf  
Set httpd user soft and hard limits as follows:  
httpd soft nofile 4096  
httpd hard nofile 10240  
Save and close the file. To see limits, enter:  
# su - httpd  
$ ulimit -Hn  
$ ulimit -Sn

Desde el punto de vista del código, este se puede ver formado por:

Secciones : Una sección esta formada por uo o mas párrafos

Parrafo: Un párrafo esta formado uno o mas sentencias

Sentencia: UN conjunto de bloques

Bloque: Un conjunto de instrucciones que se ejecutarán como una unidad; en el sentido de que, si el flujo de programa alcanza ese punto, se ejecutarán todas ellas

Instrucción: Cada uno de los verbos del lenguaje

# Manual de usuario

## Parámetros y configuración

Con excepción del componente Web, el resto de componentes del sistema se pueden ejecutar desde una consola y aceptar un conjunto de parámetros que definirán el comportamiento en esa instancia.

Estos parámetros se pueden especificar de tres maneras:

1. A través de un fichero de configuración
2. Mediante una variable de entorno
3. En la propia línea de comandos

La secuencia en la que se aplica la configuración se realiza de acuerdo con el siguiente orden:

1. Los valores por defecto establecidos en el programa
2. Las posibles variables de entorno definidas
3. Un fichero de configuración si se ha indicado en la línea de comandos
4. Las opciones establecidas de manera específica en la línea de comandos

### Fichero de configuración

El fichero de configuración es un fichero de texto que asigna a un parámetro un valor, con las siguientes características:

* Se establece una opción por línea en la forma clave = valor
* Las espacios en blanco no son significativos
* Se puede indicar un comentario con el carácter especial ‘#’ que aplica hasta el final de la línea

### Variables de entorno

Se definen a continuación las variables de entorno soportadas por el sistema

| **Variable** | **Acción** |
| --- | --- |
|  |  |
| SDP\_INPUT |  |
|  |  |
| SDP\_OUTPUT |  |
|  |  |
| SDP\_MARGIN\_LEFT |  |
|  |  |
| SDP\_MARGIN\_RIGHT |  |
|  |  |
| SDP\_QUEUE |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## SDPParser

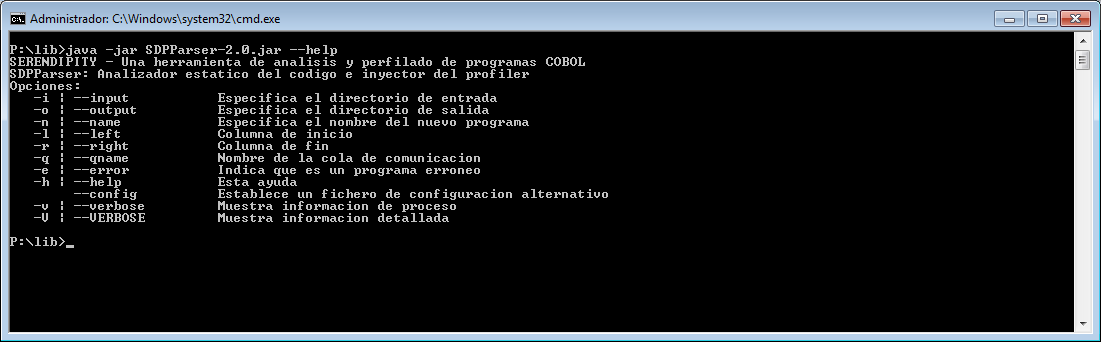
Procesa los programas COBOL pasados por línea de comandos y:

* Genera, por cada uno de ellos, otro programa funcionalmente equivalente con información de perfilado incluida
* Envía la información analítica del programa origen al servidor

Uso: java –jar SDPParser [opciones] lista\_programas\_fuente

Opciones:

|  |  |  |
| --- | --- | --- |
| -e  --error |  | Indica que se debe procesar un programa incorrecto.  Se debe utilizar cuando se ha producido un error al compilar el programa generado |
|  |  |  |
| -h  --help |  | Muestra la ayuda relativa al componente |
|  |  |  |
| -v  --verbose |  | Muestra información de progreso |
|  |  |  |
| -V |  | Muestra información más detallada del progreso |
|  |  |  |
| --config | config.properties | Especifica un fichero de configuración alternativo |
|  |  |  |
| -i  --input | directorio | Especifica el directorio de entrada por defecto donde se buscará el fichero a procesar si este no es una ruta absoluta |
|  |  |  |
| -o  --output | directorio | Especifica el directorio donde se guardará el nuevo programa generado. |
|  |  |  |
| -n  --name | nombre\_archivo | Especifica un nombre para el nuevo programa generado.  Este parámetro aplica únicamente en los casos en los que se procesa un único programa fuente |
|  |  |  |
| -l  --left | numero | Especifica donde comienza la zona A (margen izquierdo) del programa fuente |
|  |  |  |
| -r  --right | numero | Especifica la columna límite del fichero fuente |
|  |  |  |
| -q  --qname | nombre\_cola | Especifica el nombre de la cola que se usará para el envío de la información de análisis |



openSuSe 13.2 32 bits

disco swap /dev/sda1

discoduro /dev/sda2

Usuarios:

Jgonzalez373:jgg

<http://www.muycomputerpro.com/2015/03/26/demanda-expertos-cobol-universidad-formacion>

<http://www.infoweek.biz/la/2015/04/crece-demanda-expertos-cobol/>

<https://scs.senecac.on.ca/~timothy.mckenna/offline/COBOL_not_dead_yet.htm>

Guia de instalacion MQSeries

<http://www-01.ibm.com/support/docview.wss?uid=swg21656068>

# Anexos

## Configuración MQ

DEFINE PROCESS (PRC.PARSER) APPLTYPE(UNIX) APPLICID(SDPParser)

DEFINE PROCESS (PRC.TRAPPER) APPLTYPE(UNIX) APPLICID(SDPTrapper)

DEFINE PROCESS (PRC.COLLECTOR) APPLTYPE(UNIX) APPLICID(SDPCollector)

DEFINE QLOCAL (SDP.INIT) PUT(ENABLED) GET(ENABLED) +

NOTRIGGER NOSHARE

DEFINE QLOCAL (SDP.PARSER) +

PUT(ENABLED) GET(ENABLED) INITQ(SDP.INIT) +

MAXDEPTH(999999999) MAXMSGL(4194304) +

TRIGGER TRIGTYPE(FIRST) PROCESS(PRC.PARSER) +

DEFPSIST(YESY) DEFSOPT(SHARED) USAGE(NORMAL)

DEFINE QLOCAL (SDP.TRAPPER) +

PUT(ENABLED) GET(ENABLED) INITQ(SDP.INIT) +

MAXDEPTH(999999999) MAXMSGL(4194304) +

TRIGGER TRIGTYPE(FIRST) PROCESS(PRC.TRAPPER) +

DEFPSIST(YES) DEFSOPT(SHARED) USAGE(NORMAL)

DEFINE QLOCAL (SDP.COLLECTOR) +

PUT(ENABLED) GET(ENABLED) INITQ(SDP.INIT) +

MAXDEPTH(999999999) MAXMSGL(4194304) +

TRIGGER TRIGTYPE(FIRST) PROCESS(PRC.COLLECTOR) +

DEFPSIST(YES) DEFSOPT(SHARED) USAGE(NORMAL)

DEFINE QLOCAL (SDP.INIT) PUT(ENABLED) GET(ENABLED) +

NOTRIGGER NOSHARE

Lista de referencias y bibliografía

[GAR1] Gartner Blog Network . Introducing the Gartner Programming Language Index for 2014. Actualizada: 2 Octubre 2014. Disponible en: http://blogs.gartner.com/mark\_driver/2014/10/02/gartner-programming-language-index-for-2014/

[TIO1] TIOBE Index for August 2015. Disponible en:

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[CP1] Crece la demanda de expertos en COBOL pero la universidad no lo incluye en su formación. Actualizada 26 Marzo 2015. Displnible en:

http://www.muycomputerpro.com/2015/03/26/demanda-expertos-cobol-universidad-formacion

[CABE1] Thomas J. McCabe. IEEE Transactions on Software Engineering, Vol. SE-2, Nº 4, December 1976. Página 308.

Anexos

1. Definición del modelo de datos

-- -------------------------------------------------------------------

-- Definicion de tablas para MySQL

-- Autor: Javier Gonzalez

-- -------------------------------------------------------------------

DROP DATABASE SDP;

CREATE DATABASE IF NOT EXISTS SDP;

USE SDP;

SET FOREIGN\_KEY\_CHECKS = 0 ;

DROP TABLE IF EXISTS CFG\_CONFIGURACION;

CREATE TABLE CFG\_CONFIGURACION (

clave VARCHAR (127) NOT NULL

,valor VARCHAR (255) NOT NULL

,grupo INTEGER NOT NULL

,tipo INTEGER NOT NULL

,mask VARCHAR (127)

,minimo VARCHAR (32)

,maximo VARCHAR (32)

,tooltip INTEGER NOT NULL

);

ALTER TABLE CFG\_CONFIGURACION ADD PRIMARY KEY ( clave );

-- ----------------------------------------------------------

-- Areas y aplicaciones

-- ----------------------------------------------------------

DROP TABLE IF EXISTS SDP\_APLICACIONES CASCADE ;

CREATE TABLE SDP\_APLICACIONES (

aplicacion VARCHAR (32) NOT NULL

,id BIGINT UNSIGNED NOT NULL

,descripcion VARCHAR (512) NOT NULL

,padre BIGINT UNSIGNED NOT NULL

,volumen INTEGER NOT NULL

,uid VARCHAR (32) NOT NULL

,tms TIMESTAMP NOT NULL

);

ALTER TABLE SDP\_APLICACIONES ADD PRIMARY KEY ( aplicacion );

ALTER TABLE SDP\_APLICACIONES ADD UNIQUE INDEX ( id );

-- ----------------------------------------------------------

-- Lista de mascaras para asociar modulos a aplicaciones

-- ----------------------------------------------------------

DROP TABLE IF EXISTS SDP\_REL\_APP\_MOD CASCADE ;

CREATE TABLE SDP\_REL\_APP\_MOD (

idAppl BIGINT UNSIGNED NOT NULL -- Aplicacion

,mask VARCHAR (12) NOT NULL -- Mascara

,fijo INTEGER NOT NULL -- Numero de car. fijos (-1 todos)

,peso INTEGER NOT NULL -- Peso por wildcars (\* = 1000)

,uid VARCHAR (32) NOT NULL -- uid que ha generado la regla

);

ALTER TABLE SDP\_REL\_APP\_MOD ADD PRIMARY KEY ( idAppl , mask );

-- ----------------------------------------------------------

-- MODULOS

-- ----------------------------------------------------------

-- ----------------------------------------------------------

-- Definicion basica de un modulo

-- Pertenece a una aplicacion

-- ----------------------------------------------------------

DROP TABLE IF EXISTS SDP\_MODULOS CASCADE ;

CREATE TABLE SDP\_MODULOS (

idAppl BIGINT UNSIGNED NOT NULL -- Id de la aplicacion

,idModulo BIGINT UNSIGNED NOT NULL -- Id unico del modulo

,idVersion BIGINT UNSIGNED NOT NULL -- Id de la version actual

,tipo INTEGER NOT NULL -- Tipo de modulo

,nombre VARCHAR(32) NOT NULL -- Nombre del modulo

,estado INTEGER NOT NULL -- Activo (1) o Inactivo(0)

,uid VARCHAR(32) NOT NULL

,tms TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

ALTER TABLE SDP\_MODULOS ADD PRIMARY KEY ( idAppl , idModulo );

ALTER TABLE SDP\_MODULOS ADD UNIQUE INDEX ( idModulo );

-- ----------------------------------------------------------

-- Un modulo tiene varias versiones

-- ----------------------------------------------------------

DROP TABLE IF EXISTS MOD\_VERSIONES CASCADE ;

CREATE TABLE MOD\_VERSIONES (

idVersion BIGINT UNSIGNED NOT NULL -- Secuencia de la version

,idModulo BIGINT UNSIGNED NOT NULL -- Id del modulo

,firma VARCHAR (64) NOT NULL -- firma digital

,nombre VARCHAR (32) NOT NULL -- Nombre del modulo

,tipo INTEGER NOT NULL -- Tipo de modulo

,fichero INTEGER NOT NULL -- Flag tiene ficheros

,descripcion VARCHAR (1024) NOT NULL -- Posible comentario

,uid VARCHAR (32) NOT NULL -- uid que crea o actualiza

,tms TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

ALTER TABLE MOD\_VERSIONES ADD PRIMARY KEY ( idVersion );

ALTER TABLE MOD\_VERSIONES ADD UNIQUE KEY ( idModulo , idVersion);

ALTER TABLE MOD\_VERSIONES ADD UNIQUE KEY ( firma );

ALTER TABLE MOD\_VERSIONES ADD FOREIGN KEY(idModulo)

REFERENCES SDP\_MODULOS (idModulo )

ON DELETE CASCADE;

-- ----------------------------------------------------------

-- Datos resumidos de la version

-- ----------------------------------------------------------

DROP TABLE IF EXISTS MOD\_RESUMEN CASCADE ;

CREATE TABLE MOD\_RESUMEN (

idVersion BIGINT UNSIGNED NOT NULL -- Identificador de la version

,bytes BIGINT UNSIGNED NOT NULL -- Bytes del modulo

,lineas INTEGER NOT NULL -- Lineas del modulo

,parrafos INTEGER NOT NULL -- Parrafos

,sentencias INTEGER NOT NULL -- Sentencias

,blancos INTEGER NOT NULL -- Lineas en blanco

,comentarios INTEGER NOT NULL -- Lineas de comentarios

,decoradores INTEGER NOT NULL -- Comentarios decoradores

,verbosData INTEGER NOT NULL -- Sentencias

,verbosIO INTEGER NOT NULL -- Sentencias

,verbosControl INTEGER NOT NULL -- Sentencias

,verbosFlujo INTEGER NOT NULL -- Sentencias

,verbosArit INTEGER NOT NULL -- Sentencias

,verbosLang INTEGER NOT NULL -- Sentencias

,ficheros INTEGER NOT NULL -- Persistencia

);

ALTER TABLE MOD\_RESUMEN ADD PRIMARY KEY ( idVersion );

ALTER TABLE MOD\_RESUMEN ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Divisisones y secciones

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS MOD\_SECCIONES ;

CREATE TABLE MOD\_SECCIONES (

idVersion BIGINT UNSIGNED NOT NULL

,divIdentification INTEGER NOT NULL

,divEnvironment INTEGER NOT NULL

,divData INTEGER NOT NULL

,divProcedure INTEGER NOT NULL

,sectConfiguration INTEGER NOT NULL

,sectInputOutput INTEGER NOT NULL

,sectFile INTEGER NOT NULL

,sectWorking INTEGER NOT NULL

,sectLocal INTEGER NOT NULL

,sectLinkage INTEGER NOT NULL

,sectScreen INTEGER NOT NULL

);

ALTER TABLE MOD\_SECCIONES ADD PRIMARY KEY ( idVersion );

ALTER TABLE MOD\_SECCIONES ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Ficheros y tablas

-- Maestro lo puede marcar:

-- El programa

-- El administrador

-- El sistema automaticamente

-- En este caso, se elige el que mas registros haya

-- procesado por estadistica

-- Si lo ha puesto el sistema, lo cambiara si ha

-- habido otro proceso que ha leido mas registros

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS MOD\_FICHEROS ;

CREATE TABLE MOD\_FICHEROS (

idVersion BIGINT UNSIGNED NOT NULL -- Bytes del modulo

,idFile INTEGER NOT NULL -- Id del registro

,logico VARCHAR (64) NOT NULL -- Nombre logico

,fisico VARCHAR (64) NOT NULL -- Nombre fisico

,tipo INTEGER NOT NULL -- Tipo de fichero segun TP

,acceso INTEGER NOT NULL -- Tipo de acceso

,maestro INTEGER NOT NULL -- Indicador de fichero maestro

,leido BIGINT UNSIGNED NOT NULL -- Registros leidos

);

ALTER TABLE MOD\_FICHEROS ADD PRIMARY KEY ( idVersion , idFile );

ALTER TABLE MOD\_FICHEROS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- ----------------------------------------------------------

-- Dependencias

-- ----------------------------------------------------------

DROP TABLE IF EXISTS MOD\_DEPENDENCIAS ;

CREATE TABLE MOD\_DEPENDENCIAS (

idVersion BIGINT UNSIGNED NOT NULL -- Id de la verson del modulo

,modulo VARCHAR(64) NOT NULL -- Nombre del modulo o copy

,tipo INTEGER NOT NULL -- Tipo de dependencia

,subTipo INTEGER NOT NULL -- Tipo de dependencia

,estado INTEGER NOT NULL -- Verificado o no (0/1)

);

ALTER TABLE MOD\_DEPENDENCIAS ADD PRIMARY KEY ( idVersion , modulo );

ALTER TABLE MOD\_DEPENDENCIAS ADD KEY ( idVersion );

ALTER TABLE MOD\_DEPENDENCIAS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- ----------------------------------------------------------

-- Grafo

-- ----------------------------------------------------------

DROP TABLE IF EXISTS MOD\_GRAFO ;

CREATE TABLE MOD\_GRAFO (

idVersion BIGINT UNSIGNED NOT NULL -- Id de la verson del modulo

,idGrafo BIGINT UNSIGNED NOT NULL -- Id del camino

,orden BIGINT UNSIGNED NOT NULL -- Secuencia

,tipo INTEGER NOT NULL -- Tipo de nodo

,nombre VARCHAR(64) NOT NULL -- Nombre del nodo

,peso INTEGER NOT NULL -- Complejidad del nodo

);

ALTER TABLE MOD\_GRAFO ADD PRIMARY KEY ( idVersion , idGrafo, orden );

ALTER TABLE MOD\_DEPENDENCIAS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Modulo fuente original comprimido

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS MOD\_FUENTE CASCADE ;

CREATE TABLE MOD\_FUENTE (

idVersion BIGINT UNSIGNED NOT NULL -- Identificador de la version del modulo

,source BLOB NOT NULL -- Fuente en formato ZIP

);

ALTER TABLE MOD\_FUENTE ADD PRIMARY KEY ( idVersion );

ALTER TABLE MOD\_FUENTE ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Modulo fuente original comprimido de programas

-- que han fallado en el analisis

-- Se analizaran para ver la construcion que ha fallado

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS MOD\_FUENTE\_ERR CASCADE ;

CREATE TABLE MOD\_FUENTE\_ERR (

tms TIMESTAMP NOT NULL -- Timestamp de ejecución

,nombre VARCHAR (32) NOT NULL -- Nombre del modulo

,uid VARCHAR (32) NOT NULL -- uid que la ha compilado

,source BLOB NOT NULL -- Fuente en formato ZIP

);

ALTER TABLE MOD\_FUENTE\_ERR ADD PRIMARY KEY ( tms );

-- ----------------------------------------------------------

-- Bloques de codigo

-- ----------------------------------------------------------

DROP TABLE IF EXISTS MOD\_BLOQUES ;

CREATE TABLE MOD\_BLOQUES (

idVersion BIGINT UNSIGNED NOT NULL

,orden INTEGER NOT NULL -- Numero de bloque

,lineBeg INTEGER NOT NULL -- Inicio de linea

,lineEnd INTEGER NOT NULL -- Fin de linea

,sentencias INTEGER NOT NULL -- Sentencias del bloque

,usado INTEGER NOT NULL -- Se ha ejecutado (1 = si)

);

ALTER TABLE MOD\_BLOQUES ADD PRIMARY KEY ( idVersion , orden );

ALTER TABLE MOD\_BLOQUES ADD KEY ( idVersion );

ALTER TABLE MOD\_BLOQUES ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- ----------------------------------------------------------

-- Lista de parrafos

-- ----------------------------------------------------------

DROP TABLE IF EXISTS MOD\_PARRAFOS ;

CREATE TABLE MOD\_PARRAFOS (

idVersion BIGINT UNSIGNED NOT NULL

,linea INTEGER NOT NULL -- Linea del parrafo

,nombre VARCHAR(64) NOT NULL -- Nombre del parrafo

,sentencias INTEGER NOT NULL -- Sentencias del parrafo

,referencias INTEGER NOT NULL -- Veces que se referencia

,indice INTEGER NOT NULL -- Indice en la taba Working

,cc INTEGER NOT NULL -- Complejidad ciclomatica McCabe

,isExit INTEGER NOT NULL -- Es un parrafo de exit?

);

ALTER TABLE MOD\_PARRAFOS ADD PRIMARY KEY ( idVersion , linea );

ALTER TABLE MOD\_PARRAFOS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- ----------------------------------------------------------

-- Relacion entre parrafos

-- ----------------------------------------------------------

DROP TABLE IF EXISTS REL\_MOD\_PARRAFOS ;

CREATE TABLE REL\_MOD\_PARRAFOS (

idVersion BIGINT UNSIGNED NOT NULL

,id BIGINT UNSIGNED NOT NULL -- Id Unico.

-- Un parrafo A puede llamar a otro B varias veces

,origen BIGINT UNSIGNED NOT NULL -- Linea del parrafo origen

,destino BIGINT UNSIGNED NOT NULL -- Linea del parrafo destino

);

ALTER TABLE REL\_MOD\_PARRAFOS ADD PRIMARY KEY ( idVersion , id );

ALTER TABLE REL\_MOD\_PARRAFOS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- TABLAS CARGADAS COMO RESULTADO DE UNA EJECUCION

-- PREFIJO: TRP

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS TRP\_SESIONES ;

CREATE TABLE TRP\_SESIONES (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idModulo VARCHAR(64) NOT NULL -- Id del modulo principal

,elapsed BIGINT UNSIGNED NOT NULL -- Elapsed time

,cpu BIGINT UNSIGNED NOT NULL -- CPU time

,suspend BIGINT UNSIGNED NOT NULL -- Suspend time

,leido BIGINT UNSIGNED NOT NULL -- Registros leidos

,escrito BIGINT UNSIGNED NOT NULL -- Registros escritos

,finished INTEGER NOT NULL -- Ha finalizado el proceso?

,uid VARCHAR(32) NOT NULL -- uid que lo ha ejecutado

,tms TIMESTAMP NOT NULL -- Timestamp de ejecución

);

ALTER TABLE TRP\_SESIONES ADD PRIMARY KEY ( idSesion, idModulo );

ALTER TABLE TRP\_SESIONES ADD FOREIGN KEY ( idModulo )

REFERENCES MOD\_VERSIONES ( firma )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Modulos invocados en la sesion incluyendo el modulo principal

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS TRP\_MODULOS ;

CREATE TABLE TRP\_MODULOS (

idSesion VARCHAR (64) NOT NULL -- Id de la sesion

,idModulo VARCHAR (64) NOT NULL -- Id de la version

,orden BIGINT UNSIGNED NOT NULL -- Contador de mensajes

,modulo VARCHAR (64) NOT NULL -- Nombre del modulo

,veces BIGINT UNSIGNED NOT NULL -- Numero de ocurrencias

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo wait/suspend

);

ALTER TABLE TRP\_MODULOS ADD PRIMARY KEY ( idSesion , idModulo );

ALTER TABLE TRP\_MODULOS ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- ----------------------------------------------------------

-- Registros de cobertura

-- Una sesion puede afectar a varios modulos

-- ----------------------------------------------------------

DROP TABLE IF EXISTS TRP\_COBERTURA ;

CREATE TABLE TRP\_COBERTURA (

idSesion VARCHAR(64) NOT NULL -- MD5 de la sesion

,idModulo VARCHAR(64) NOT NULL -- MD5 de la sesion

,orden BIGINT UNSIGNED NOT NULL -- Numero de mensaje

,flags VARCHAR(4000) NOT NULL -- Tabla working con flags de uso

);

ALTER TABLE TRP\_COBERTURA ADD PRIMARY KEY ( idSesion, idModulo );

ALTER TABLE TRP\_COBERTURA ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Registros de persistencia

-- Una sesion puede ejecutar varias veces el mismo modulo

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS TRP\_PERSISTENCIA ;

CREATE TABLE TRP\_PERSISTENCIA (

idSesion VARCHAR(64) NOT NULL -- MD5 de la sesion

,idModulo VARCHAR(64) NOT NULL -- MD5 del modulo

,orden BIGINT UNSIGNED NOT NULL -- Numero del mensaje

,flags VARCHAR(4000) NOT NULL -- Flags de bloques

);

ALTER TABLE TRP\_PERSISTENCIA ADD PRIMARY KEY ( idSesion, idModulo);

ALTER TABLE TRP\_PERSISTENCIA ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- ----------------------------------------------------------

-- Registros de llamadas a parrafos

-- Datos obtenidos de la working

-- ----------------------------------------------------------

DROP TABLE IF EXISTS TRP\_PARR\_WORKING ;

CREATE TABLE TRP\_PARR\_WORKING (

idSesion VARCHAR(64) NOT NULL -- MD5 de la sesion

,idModulo VARCHAR(64) NOT NULL -- MD5 del modulo

,orden BIGINT UNSIGNED NOT NULL -- Numero de mensaje

,datos VARCHAR(4095) NOT NULL -- Tabla working de contadores

);

ALTER TABLE TRP\_PARR\_WORKING ADD PRIMARY KEY ( idSesion, idModulo );

ALTER TABLE TRP\_PARR\_WORKING ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Registros de llamadas a parrafos

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS TRP\_PERFORM ;

CREATE TABLE TRP\_PERFORM (

idSesion VARCHAR(64) NOT NULL -- MD5 de la sesion

,idModulo VARCHAR(64) NOT NULL -- MD5 del modulo

,orden BIGINT UNSIGNED NOT NULL -- Numero del mensaje

,etiqueta VARCHAR(64) NOT NULL -- Etiqueta: modulo, parrafo, nombre

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

);

ALTER TABLE TRP\_PERFORM ADD PRIMARY KEY ( idSesion , idModulo, etiqueta , orden );

ALTER TABLE TRP\_PERFORM ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Registros de uso de parrafos

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS TRP\_PARRAFOS ;

CREATE TABLE TRP\_PARRAFOS (

idSesion VARCHAR(64) NOT NULL -- MD5 de la sesion

,idModulo VARCHAR(64) NOT NULL -- MD5 del modulo

,orden BIGINT UNSIGNED NOT NULL -- Numero del mensaje

,etiqueta VARCHAR(64) NOT NULL -- Nombre del parrafo

,veces BIGINT UNSIGNED NOT NULL -- Numero de instancias

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

);

ALTER TABLE TRP\_PARRAFOS ADD PRIMARY KEY ( idSesion , idModulo, etiqueta , orden );

ALTER TABLE TRP\_PARRAFOS ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Registros de llamadas a modulos

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS TRP\_CALL ;

CREATE TABLE TRP\_CALL (

idSesion VARCHAR(64) NOT NULL -- MD5 de la sesion

,orden BIGINT UNSIGNED NOT NULL -- Numero del mensaje

,idCalling VARCHAR(64) NOT NULL -- MD5 del modulo llamante

,idCalled VARCHAR(64) NOT NULL -- MD5 del modulo llamado

,modulo VARCHAR(64) NOT NULL -- Nombre de la rutina llamada

,veces BIGINT UNSIGNED NOT NULL -- Numero de instancias

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

);

ALTER TABLE TRP\_CALL ADD PRIMARY KEY ( idSesion, orden, idCalling, idCalled );

ALTER TABLE TRP\_CALL ADD FOREIGN KEY ( idSesion )

REFERENCES TRP\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- TABLAS SUMARIZADAS

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- TABLAS SUMARIZADAS DE LAS EJECUCIONES

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- Ejecuciones

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SES\_SESIONES ;

CREATE TABLE SES\_SESIONES (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idVersion BIGINT UNSIGNED NOT NULL -- Id del modulo principal

,elapsed BIGINT UNSIGNED NOT NULL -- Elapsed time

,cpu BIGINT UNSIGNED NOT NULL -- CPU time

,suspend BIGINT UNSIGNED NOT NULL -- Suspend time

,leido BIGINT UNSIGNED NOT NULL -- Registros procesados

,escrito BIGINT UNSIGNED NOT NULL -- Registros generados

,finished INTEGER NOT NULL -- Ha finalizado el proceso?

,uid VARCHAR(32) NOT NULL -- uid que lo ha ejecutado

,tms TIMESTAMP NOT NULL -- Timestamp de ejecución

);

ALTER TABLE SES\_SESIONES ADD PRIMARY KEY ( idSesion, idVersion );

ALTER TABLE SES\_SESIONES ADD KEY ( idVersion );

ALTER TABLE SES\_SESIONES ADD KEY ( tms );

ALTER TABLE SES\_SESIONES ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Persistencia

-- En una misma sesion pueden existir varias ejecuciones

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SES\_PERSISTENCIA ;

CREATE TABLE SES\_PERSISTENCIA (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idVersion BIGINT UNSIGNED NOT NULL -- Id de la version modulo

,orden BIGINT UNSIGNED NOT NULL -- Cuenta de ejecuciones

,idFile INTEGER NOT NULL -- Id del fichero

,maestro INTEGER NOT NULL -- Indicador de maestro

,acOpen BIGINT UNSIGNED NOT NULL -- Apertura

,acClose BIGINT UNSIGNED NOT NULL -- Cierre

,acInsert BIGINT UNSIGNED NOT NULL -- Insert / Write

,acRead BIGINT UNSIGNED NOT NULL -- Read

,acUpdate BIGINT UNSIGNED NOT NULL -- Update / Rewrite

,acDelete BIGINT UNSIGNED NOT NULL -- Delete

,acSelect BIGINT UNSIGNED NOT NULL -- Select

,acTotal BIGINT UNSIGNED NOT NULL -- Total de registros

);

ALTER TABLE SES\_PERSISTENCIA ADD PRIMARY KEY ( idSesion , idVersion , orden , idFile);

ALTER TABLE SES\_PERSISTENCIA ADD FOREIGN KEY ( idSesion )

REFERENCES SES\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Acumulado de entradas/salidas realizadas

-- Total de registros leidos escritos etc en la sesion

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SES\_IO ;

CREATE TABLE SES\_IO (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idVersion BIGINT UNSIGNED NOT NULL -- Id de la version modulo

,acOpen BIGINT UNSIGNED NOT NULL -- Apertura

,acClose BIGINT UNSIGNED NOT NULL -- Cierre

,acInsert BIGINT UNSIGNED NOT NULL -- Insert / Write

,acRead BIGINT UNSIGNED NOT NULL -- Read

,acUpdate BIGINT UNSIGNED NOT NULL -- Update / Rewrite

,acDelete BIGINT UNSIGNED NOT NULL -- Delete

,acSelect BIGINT UNSIGNED NOT NULL -- Select

,acTotal BIGINT UNSIGNED NOT NULL -- Total de registros

,tms TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP

);

ALTER TABLE SES\_IO ADD PRIMARY KEY ( idSesion );

ALTER TABLE SES\_IO ADD KEY ( idVersion , tms );

ALTER TABLE SES\_IO ADD FOREIGN KEY ( idSesion )

REFERENCES SES\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Tiempos consumidos en cada parrafo de cada modulo por sesion

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SES\_PARRAFOS ;

CREATE TABLE SES\_PARRAFOS (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idVersion BIGINT UNSIGNED NOT NULL -- Id de la version modulo

,orden BIGINT UNSIGNED NOT NULL -- Varios modulos por sesion

,nombre VARCHAR(64) NOT NULL -- Etiqueta: modulo, parrafo, nombre

,veces BIGINT UNSIGNED NOT NULL -- Numero de veces que se ha ejecutado

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,avgTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,avgIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,avgIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,minTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,minTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,minTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,minIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,minIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,minIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,maxTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,maxIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,maxIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque

);

ALTER TABLE SES\_PARRAFOS ADD PRIMARY KEY ( idSesion, idVersion, orden, nombre );

ALTER TABLE SES\_PARRAFOS ADD FOREIGN KEY ( idSesion )

REFERENCES SES\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Arbol de llamadas de la sesion

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SES\_ARBOL ;

CREATE TABLE SES\_ARBOL (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idCalling BIGINT UNSIGNED NOT NULL -- Id de la version del modulo llamante

,idCalled BIGINT UNSIGNED NOT NULL -- Id de la version del modulo llamado (0 si no se monitoriza)

,nombre VARCHAR(32) NOT NULL -- Nombre del modulo llamado

,veces BIGINT UNSIGNED NOT NULL -- Numero de veces que se ha ejecutado

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,tms TIMESTAMP NOT NULL

);

ALTER TABLE SES\_ARBOL ADD PRIMARY KEY ( idSesion, idCalling, idCalled);

ALTER TABLE SES\_ARBOL ADD FOREIGN KEY ( idCalling )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Modulos y tiempos consumidos en la sesion

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SES\_MODULOS ;

CREATE TABLE SES\_MODULOS (

idSesion VARCHAR(64) NOT NULL -- Id de la sesion

,idVersion BIGINT UNSIGNED NOT NULL -- Id de la version modulo

,orden BIGINT UNSIGNED NOT NULL -- Varios modulos por sesion

,nombre VARCHAR(64) NOT NULL -- Nombre del modulo

,veces BIGINT UNSIGNED NOT NULL -- Numero de veces que se ha ejecutado

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,avgTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,avgIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,avgIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,minTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,minTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,minTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,minIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,minIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,minIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,maxTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,maxIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,maxIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,tms TIMESTAMP NOT NULL -- Timestamp de la sesion

);

ALTER TABLE SES\_MODULOS ADD PRIMARY KEY ( idSesion, idVersion, orden );

ALTER TABLE SES\_MODULOS ADD FOREIGN KEY ( idSesion )

REFERENCES SES\_SESIONES ( idSesion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- TABLAS SUMARIZADAS.

-- MEDIAS TOTALES

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SUM\_MODULOS ;

CREATE TABLE SUM\_MODULOS (

idVersion BIGINT UNSIGNED NOT NULL -- Id de la version modulo

,idModulo BIGINT UNSIGNED NOT NULL -- Id del modulo

,nombre VARCHAR(64) NOT NULL -- Nombre del modulo

,veces BIGINT UNSIGNED NOT NULL -- Numero de veces que se ha ejecutado

,avgElapsed BIGINT UNSIGNED NOT NULL -- Media de tiempo

,avgCpu BIGINT UNSIGNED NOT NULL -- Media de CPU

,avgSuspend BIGINT UNSIGNED NOT NULL -- Media de suspend

,minElapsed BIGINT UNSIGNED NOT NULL -- Minimo de tiempo

,minCpu BIGINT UNSIGNED NOT NULL -- Minimo de CPU

,minSuspend BIGINT UNSIGNED NOT NULL -- Minimo de suspend

,maxElapsed BIGINT UNSIGNED NOT NULL -- Minimo de tiempo

,maxCpu BIGINT UNSIGNED NOT NULL -- Maximo de CPU

,maxSuspend BIGINT UNSIGNED NOT NULL -- Maximo de Suspend

,avgLeido BIGINT UNSIGNED NOT NULL -- Media de registros leidos

,avgEscrito BIGINT UNSIGNED NOT NULL -- Media de registros escritos

,minLeido BIGINT UNSIGNED NOT NULL -- Minimo de registros leidos

,minEscrito BIGINT UNSIGNED NOT NULL -- Minimo de registros escritos

,maxLeido BIGINT UNSIGNED NOT NULL -- Maximo de registros leidos

,maxEscrito BIGINT UNSIGNED NOT NULL -- Maximo de registros escritos

);

ALTER TABLE SUM\_MODULOS ADD PRIMARY KEY ( idVersion );

ALTER TABLE SUM\_MODULOS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Tiempos consumidos en cada parrafo de cada modulo por sesion

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SUM\_PARRAFOS ;

CREATE TABLE SUM\_PARRAFOS (

idVersion BIGINT UNSIGNED NOT NULL -- Id de la version modulo

,nombre VARCHAR(64) NOT NULL -- Etiqueta: modulo, parrafo, nombre

,sesiones BIGINT UNSIGNED NOT NULL -- Numero de sesiones

,veces BIGINT UNSIGNED NOT NULL -- Numero de veces que se ha ejecutado

,linea INTEGER NOT NULL -- Linea dentro del modulo fuente

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,intElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,intCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,intSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,avgTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,avgIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,avgIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,minTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,minTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,minTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,minIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,minIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo uid

,minIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque en modo kernel

,maxTotElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxTotCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxTotSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,maxIntElapsed BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,maxIntCpu BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,maxIntSuspend BIGINT UNSIGNED NOT NULL -- Tiempo bloque

,tms TIMESTAMP NOT NULL

);

ALTER TABLE SUM\_PARRAFOS ADD PRIMARY KEY ( idVersion, nombre );

ALTER TABLE SUM\_PARRAFOS ADD FOREIGN KEY ( idVersion )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Arbol de llamadas

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS SUM\_ARBOL ;

CREATE TABLE SUM\_ARBOL (

idCalling BIGINT UNSIGNED NOT NULL -- Id de la version del modulo llamante

,idCalled BIGINT UNSIGNED NOT NULL -- Id de la version del modulo llamado (0 si no se monitoriza)

,nombre VARCHAR(32) NOT NULL -- Nombre del modulo llamado

,sesiones BIGINT UNSIGNED NOT NULL -- Numero de sesiones monitorizadas

,veces BIGINT UNSIGNED NOT NULL -- Numero de veces que se ha ejecutado

,totElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total

,totCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,totSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgElapsed BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgCpu BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,avgSuspend BIGINT UNSIGNED NOT NULL -- Tiempo total en modo uid

,tms TIMESTAMP NOT NULL

);

ALTER TABLE SUM\_ARBOL ADD PRIMARY KEY ( idCalling, idCalled, nombre );

ALTER TABLE SUM\_ARBOL ADD FOREIGN KEY ( idCalling )

REFERENCES MOD\_VERSIONES ( idVersion )

ON DELETE CASCADE ;

-- -------------------------------------------------------------------

-- Tablas de codigos

-- Contiene las descripciones de los codigos localizadas

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS CFG\_CODIGOS ;

CREATE TABLE CFG\_CODIGOS (

grupo INTEGER NOT NULL -- Grupo de codigos

,codigo INTEGER NOT NULL -- Identificador

,lang VARCHAR(02) NOT NULL -- Codigo de idioma

,valor VARCHAR(64) NOT NULL -- Descripcion

);

ALTER TABLE CFG\_CODIGOS ADD PRIMARY KEY ( grupo , codigo , lang);

-- -------------------------------------------------------------------

-- Tablas de logging

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS LOG\_LOGGING ;

CREATE TABLE LOG\_LOGGING (

secuencia BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY -- Clave unica

,idAppl BIGINT UNSIGNED NOT NULL -- Identificador de la aplicacion

,idModulo BIGINT UNSIGNED NOT NULL -- Identificador del modulo

,idVersion BIGINT UNSIGNED NOT NULL -- Identificador de la version

,idTipo INTEGER NOT NULL -- Identificador del tipo de mensaje

,idMsg INTEGER NOT NULL -- Identificador del mensaje

,uid VARCHAR(32) NOT NULL -- uid

,objeto VARCHAR(64) NOT NULL -- Nombre del objeto

,tms TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP

-- Campos para rellenar el mensaje

,dato0 VARCHAR(64) -- Campo 0

,dato1 VARCHAR(64) -- Campo 2

,dato2 VARCHAR(64) -- Campo 3

,dato3 VARCHAR(64) -- Campo 3

,dato4 VARCHAR(64) -- Campo 4

,dato5 VARCHAR(64) -- Campo 5

,dato6 VARCHAR(64) -- Campo 6

,dato7 VARCHAR(64) -- Campo 7

,dato8 VARCHAR(64) -- Campo 8

,dato9 VARCHAR(64) -- Campo 9

);

ALTER TABLE LOG\_LOGGING ADD INDEX ( tms DESC );

ALTER TABLE LOG\_LOGGING ADD INDEX ( idTipo );

ALTER TABLE LOG\_LOGGING ADD INDEX ( tms DESC, idMsg );

ALTER TABLE LOG\_LOGGING ADD INDEX ( idAppl , idModulo );

ALTER TABLE LOG\_LOGGING ADD INDEX ( uid );

-- -------------------------------------------------------------------

-- Tablas de logging

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS LOG\_MSG ;

CREATE TABLE LOG\_MSG (

idMsg INTEGER NOT NULL -- Identificador del mensaje

,lang CHAR(2) NOT NULL -- Codigo de idioma

,msg VARCHAR(255) NOT NULL -- Formato del mensaje

);

ALTER TABLE LOG\_MSG ADD PRIMARY KEY ( idMsg , lang );

-- -------------------------------------------------------------------

-- Tablas de logging

-- -------------------------------------------------------------------

DROP TABLE IF EXISTS LOG\_TIPO ;

CREATE TABLE LOG\_TIPO (

idTipo INTEGER NOT NULL -- Identificador del mensaje

,lang CHAR(2) NOT NULL -- Codigo de idioma

,msg VARCHAR(255) NOT NULL -- Formato del mensaje

);

ALTER TABLE LOG\_TIPO ADD PRIMARY KEY ( idTipo , lang );

SET FOREIGN\_KEY\_CHECKS = 1 ;

COMMIT;

1. Definición del sistema de mensajería

DEFINE PROCESS (PRC.PARSER) APPLTYPE(UNIX) APPLICID(SDPParser)

DEFINE PROCESS (PRC.TRAPPER) APPLTYPE(UNIX) APPLICID(SDPTrapper)

DEFINE PROCESS (PRC.COLLECTOR) APPLTYPE(UNIX) APPLICID(SDPCollector)

DEFINE QLOCAL (SDP.INIT) PUT(ENABLED) GET(ENABLED) +

NOTRIGGER NOSHARE

DEFINE QLOCAL (SDP.PARSER) +

PUT(ENABLED) GET(ENABLED) INITQ(SDP.INIT) +

MAXDEPTH(999999999) MAXMSGL(4194304) +

TRIGGER TRIGTYPE(FIRST) PROCESS(PRC.PARSER) +

DEFPSIST(YESY) DEFSOPT(SHARED) USAGE(NORMAL)

DEFINE QLOCAL (SDP.TRAPPER) +

PUT(ENABLED) GET(ENABLED) INITQ(SDP.INIT) +

MAXDEPTH(999999999) MAXMSGL(4194304) +

TRIGGER TRIGTYPE(FIRST) PROCESS(PRC.TRAPPER) +

DEFPSIST(YES) DEFSOPT(SHARED) USAGE(NORMAL)

DEFINE QLOCAL (SDP.COLLECTOR) +

PUT(ENABLED) GET(ENABLED) INITQ(SDP.INIT) +

MAXDEPTH(999999999) MAXMSGL(4194304) +

TRIGGER TRIGTYPE(FIRST) PROCESS(PRC.COLLECTOR) +

DEFPSIST(YES) DEFSOPT(SHARED) USAGE(NORMAL)

DEFINE QLOCAL (SDP.INIT) PUT(ENABLED) GET(ENABLED) +

NOTRIGGER NOSHARE

1. Script de compilación

#!/bin/sh

############################################################

# Utilidad para compilar un programa COBOL y analizarlo

#

# Autor : Javier Gonzalez Grandez

# Fecha : Ago - 2015

# Version: 1.0

############################################################

############################################################

# Variables globales

############################################################

TMPDIR=/tmp

OUTEXE=/PFC/cases/exe

OUTDLL=/PFC/cases/shared

VERBOSE=0

############################################################

# Funciones

############################################################

showHelp() {

echo "Utilidad para compilar y preparar programas COBOL "

echo " "

echo "Uso: `basename $0` [opciones] lista\_de\_programas\_fuente "

echo "Opciones: "

echo " -o | --out Directorio de salida "

echo " -h | --help Muestra esta ayuda "

echo " -v | --verbose Muestra informacion de progreso "

exit 0

}

compila() {

SRC=`basename $2`

BASE=${SRC%%.\*}

if [ $VERBOSE -gt 0 -a $4 -eq 0 ] ; then

echo -n "Compilando $BASE "

fi

if [ $1 -eq 0 ] ; then

OPT=-x

EXT=exe

else

OPT=-m

EXT=so

fi

echo cobc $OPT -o $3/${BASE}.${EXT} $2

cobc $OPT -o ${3}/${BASE}.${EXT} $2

return $?

}

############################################################

# Code

############################################################

OPTS=$(getopt -n "$0" -o o:vh --long "out:,verbose,help" -- "$@")

if [ $? -ne 0 ]; then

echo An unexpected error was occured

exit 9

fi

eval set -- "$OPTS"

while true;

do

case "$1" in

-h | --help)

showHelp

shift

;;

-v | --verbose)

VERBOSE=1

shift

;;

-o | --out)

if [ -n "$2" ] ; then

OUTDIR=$2

else

echo "ERROR: Falta el directorio de salida"

exit 1

fi

shift 2

;;

--) shift

break

;;

esac

done

# Compilar los programas

for src in $\* ; do

SRC=`basename $src`

DLL=0

# Identificar si es programa o rutina

egrep -qi "DIVISION[ \t]+USING" $src

if [ $? -eq 0 ] ; then

DLL=1

fi

compila $DLL $src $TMPDIR 0

rc=$?

if [ $rc -ne 0 ] ; then

if [$VERBOSE -gt 0 ] ; then

echo " - KO"

echo "ERROR: El programa fuente no es correcto"

fi

exit $rc

fi

java -jar SDPParser.jar -n ${TMPDIR}/${SRC} $src

rc=$?

if [ $rc -ne 0 ] ; then

if [$VERBOSE -gt 0 ] ; then

echo " - KO"

echo "ERROR: Se ha producido un error al analizar el programa"

fi

exit $rc

fi

# Establece el directorio de salida

if [ -z "$OUTDIR" ] ; then

if [ $DLL -eq 1 ] ; then

OUTDIR=$OUTDLL

else

OUTDIR=$OUTEXE

fi

fi

compila $DLL ${TMPDIR}/${SRC} $OUTDIR 1

rc=$?

# Guardar el programa erroneo para su analisis

if [ $rc -ne 0 ] ; then

java -jar SDPParser.jar -e $src

exit $rc

fi

done

1. Programas de prueba

| Prefijo |  | Acción |
| --- | --- | --- |
|  |  |  |
| BAD\* | 01\* | Contiene sentencias no permitidas (GO TO) |
|  |  |  |
| vsftpd |  | Servidor ftp |
|  |  |  |
| gcc |  | Compilador GNU C |
|  |  |  |
| rpm |  | Gestor de paquetes RPM |
|  |  |  |
| Make |  | Utilidad para gestión de compilaciones |
|  |  |  |
| automake |  | Herramienta para la generación de Makefiles conformes con los estándares GNU |
|  |  |  |
| autoconf |  | Herramienta de generación automática de scripts de configuración |
|  |  |  |
| default-jre |  | Open Java Runtime Environment |
|  |  |  |
| default-jdk |  | Open Java Development Toolkit |
|  |  |  |
| open-cobol |  | Compilador Open Cobol |
|  |  |  |
| apache2 |  | Servidor de páginas Apache 2 |
|  |  |  |
| tomcat8 |  | Servidor de aplicaciones Apache Tomcat 8 |
|  |  |  |
| tomcat8-admin |  | Aplicación de administración Web de Apache Tomcat 8 |
|  |  |  |
| tomcat8-user |  | Herramienta para crear instancias de usuarios en Apache Tomcat |
|  |  |  |
| mysql-server |  | Servidor de base de datos MySQL |
|  |  |  |

1. Software de terceras partes

| Prefijo |  | Acción |
| --- | --- | --- |
|  |  |  |
| BAD\* | 01\* | Contiene sentencias no permitidas (GO TO) |
|  |  |  |
| vsftpd |  | Servidor ftp |
|  |  |  |
| gcc |  | Compilador GNU C |
|  |  |  |
| rpm |  | Gestor de paquetes RPM |
|  |  |  |
| Make |  | Utilidad para gestión de compilaciones |
|  |  |  |
| automake |  | Herramienta para la generación de Makefiles conformes con los estándares GNU |
|  |  |  |
| autoconf |  | Herramienta de generación automática de scripts de configuración |
|  |  |  |
| default-jre |  | Open Java Runtime Environment |
|  |  |  |
| default-jdk |  | Open Java Development Toolkit |
|  |  |  |
| open-cobol |  | Compilador Open Cobol |
|  |  |  |
| apache2 |  | Servidor de páginas Apache 2 |
|  |  |  |
| tomcat8 |  | Servidor de aplicaciones Apache Tomcat 8 |
|  |  |  |
| tomcat8-admin |  | Aplicación de administración Web de Apache Tomcat 8 |
|  |  |  |
| tomcat8-user |  | Herramienta para crear instancias de usuarios en Apache Tomcat |
|  |  |  |
| mysql-server |  | Servidor de base de datos MySQL |
|  |  |  |