TEMA 4. Herramientas para el mantenimiento de los sistemas informáticos

(Duración aproximada: 2 horas)

4.1.- Introducción.

- 4.1.1.- Herramientas para el mantenimiento.
- 4.1.2.- Según concepción.
  - 4.1.2.1.- Específicas.
  - 4.1.2.2.- Generales.
- 4.1.3.- Según nivel de aplicación.
  - 4.1.3.1.- Instrumentos.
  - 4.1.3.2.- Equipos.
  - 4.1.3.3.- Sistemas.

4.2.- Instrumentación.

- 4.2.1.- Características.
- 4.2.2.- Tipos de instrumentación.
  - 4.2.2.1.- Analógica.
  - 4.2.2.2.- Digital.
  - 4.2.2.3.- Mixta.
  - 4.2.2.4.- Comunicaciones.
- 4.2.3.- Estándares.
  - 4.2.3.1.- Estándares de test.
  - 4.2.3.2.- Estándares de comunicación.

4.3.- Herramientas lógicas (software).

- 4.3.1.- Características.
- 4.3.2.- Tipos.
  - 4.3.2.1.- Depuración de hardware.
  - 4.3.2.2.- Depuración de software.
  - 4.3.2.3.- Aplicaciones mixtas.
  - 4.3.2.4.- Otros.

4.4.- Herramientas para la gestión del mantenimiento.

1. Introducción.

Este capítulo presenta algunas nociones básicas acerca de las herramientas utilizadas en el manteniminto de sistemas, principalmente en el mantenimiento de sistemas informáticos. Se dan algunas clasificaciones interesantes, ejemplos y buenas prácticas de mantenimiento en el abito de la programación.

1.1. Herramientas para el mantenimiento.

El conjunto de herramientas que se pueden emplear en el mantenimiento de sistemas es tan amplio o más que la diversidad misma de los sistemas susceptibles de ser mantenidos. Incluso, de forma recursiva, buena parte de las herramientas utilizadas en el mantenimiento requieren a su vez de mantenimiento para continuar siendo operativas. Por citar un ejemplo curioso, la herramienta más adecuada para identificar un problema en un analizador lógico podría ser, según el caso, otro analizador lógico.

De este modo, incluso restringiéndonos a los sistemas informaticos, la diversidad de las herramientas y la complejidad de las más especializadas hacen que intentar describirlas o simplemente presentarlas todas sea una tarea imposible. Desde elementos tan simples como unas pinzas, un destornillador o un soldador para reparar circuitos o cableados hasta una cámara blanca para la fabricación de una nueva versión de un circuito integrado, pasando por un sistema JTAG compuesto de hardware específico, software y un ordenador, todo son herramientas válidas para mantener sistemas electrónicos digitales.

En el caso de los sistemas informáticos –sobre todo, aunque no solamente, en los ordenadores– se da además la circunstancia de que tanto el sistema a mantener como la herramienta pueden ser *hardware*, *software* –y recordemos que el software no tiene existencia física– o una mezcla de ambos, lo que da todavía mayor diversidad a las herramientas.

Como ejercicio que ejemplifica esta última frase es interesante intentar encontrar –si acaso existen– , dentro del mantenimiento de ordenadores:

- *herramientas hardware para mantener hardware,*
- *herramientas software para mantener software,*
- *herramientas hardware para mantener software,*
- *herramientas software para mantener hardware,*
- *. . .*

El presente tema pues se dedica a dar una visión global de las herramientas que se aplican al mantenimiento, centrándose en el mantenimiento de sistemas informáticos y en las herramientas de diagnóstico de averías –la fase comunmente más difícil del mantenimiento correctivo. Se presentan algunas clasificaciones, ejemplos y técnicas. Al final se incluye un apartado acerca de las herramientas informáticas –aplicaciones– que ayudan a la gestión del mantenimiento en general.

1.2. Según concepción.

Como se vio en el tema uno, durante las fases de estudio y producción del ciclo de vida de las instalaciones se pueden desarrollar herramientas que se vayan a utilizar en su mantenimiento. Sin embargo hemos citado antes una serie de herramientas para el mantenimiento que no es necesario diseñar ni fabricar puesto que son de uso general o válidas para un amplio conjunto de sistemas. Atendiendo a si las herramientas se deben crear –concebir– *ad hoc* para un sistema o son de aplicación más genérica, se pueden clasificar como

- específicas o
- generales.

1.2.1. Específicas.

Son aquellas herramientas que se diseñan para una instalación o sistema. Su uso es privativo de y suele estar limitado al sistema para el cual se han diseñado, siendo por ello muy potentes y eficaces en su mantenimiento.

Como contrapartida, su coste es elevado y su uso acaba cuando termina el ciclo de vida del sistema al cual se aplican. Están pues justificadas únicamente para sistemas de elevado coste que no puedan ser mantenidos de forma comparable con herramientas genéricas.

1.2.2. Generales.

Son herramientas cuyo campo de aplicación incluye todo tipo de sistemas o una clase suficientemente grande de estos. Se diseñan y fabrican pensando en una aplicabilidad general y no en un sistema en particular. La totalidad de las herramientas que se usarán como ejemplos en las siguientes secciones pertenecen a este tipo.

Hoy en día, debido al extendido uso de los sistemas empotrados y a la aplicación de estándares en muchos ámbitos de la industria, las herramientas específicas se diseñan utilizando componentes más genéricos y software de personalización y configuración, reduciendo sensiblemente su coste y su tiempo de desarrollo. La introducción de estándares en el diseño de los sistemas, muchos de los cuales tienen que ver con su mantenibilidad, permite que las herramientas puedan ser diseñadas conociendo a priori las especificaciones físicas, eléctricas, de protocolo, etcétera de los sistemas a mantener. De este modo los sistemas empotrados que constituirán las herramientas pueden incluir los componentes *hardware* necesarios para comunicarse mediante estos estándares y el *software* que interprete los datos obtenidos sin que el sistema a mantener haya sido fabricado todavía. Finalmente el caracter específico lo dará una capa de configuración para adaptarse al sistema y posiblemente la interfaz de usuario. Un ejemplo común de este tipo de herramientas se puede encontrar en los sistemas de diagnostico de averías en vehículos.

1.3. Según nivel de aplicación.

Si atendemos a la extensión del sistema sobre el que se pueden aplicar las herramientas, éstas se pueden clasificar en

- instrumentos,
- equipos y
- sistemas.

1.3.1. Instrumentos.

Se trata de herramientas de extensión reducida, que tienen acceso local a un sólo elemento de la instalación que se está manteniendo y de la cual pueden tomar un número limitado de muestras o medidas. Ejemplos simples del ámbito del mantenimiento de sistemas electrónicos serían los polímetros, osciloscopios, etcétera.

1.3.2. Equipos.

Son conjuntos de herramientas o herramientas múltiples que permiten tomar muestras de un sistema completo o de varios sistemas suficientemente próximos. Las medidas así tomadas deben estar sincronizadas o coordinadas de alguna manera. Ejemplos serían un analizador mixto –osciloscopio y analizador lógico en un mismo bastidor de sistema, el uso de *software* específico y un osciloscopio para depurar un sistema de comunicación RS232 ...

1.3.3. Sistemas.

Un sistema está formado por un conjunto de equipos interconectados y sincronizados entre sí que permite monitorizar una instalación completa. Ejemplos de sistemas los tenemos en los conjuntos de instrumentos conectados por un bus GPIB, en las aplicaciones que monitorizan coordinadamente diversos ordenadores en red, etcétera.

2. Instrumentación.

Se ha comentado en la introducción que el tema se va a ocupar fundamentalmente de las herramientas de toma de muestras o de medida ya que son necesarias para el diagnóstico de averías, que es la parte más complicada en toda intervención de mantenimento correctivo. En los sistemas informaticos, electrónicos y eléctricos en general, esta clase de herramientas es comúnmente conocida como instrumentacion.

2.1. Características.

La mayor parte de las herramientas actuales son sistemas empotrados, es decir, sistemas electrónicos digitales. De esta manera su estructura más general suele ser la siguiente:

- Un elemento transductor o acondicionador de señal –en el caso frecuente de medir magnitudes eléctricas– que transmita con la menor distorsión posible la magnitud a medir transformada mediante alguna función conocida al siguiente elemento de la entrada de señal. Debe ser poco instrusivo, es decir, afectar lo menos posible al propio sistema que se está midiendo.
- Un elemento conversor –para señales analógicas– o un detector de umbral –para señales digitales– que permita generar el valor que el sistema de medida asigna a la magnitud monitorizada. La velocidad de adquisición y la precisión –resolución, en el caso de los conversores– son los factores que determinan la calidad de esta etapa del sistema.
- Dispositivos de lectura, procesado y almacenamiento de los datos obtenidos. La capacidad de almacenamiento es fundamental en los sitemas que deben ser capaces de adquirir muestras a una gran velocidad.
- Una interfaz de usuario normalmente más parecida a la que ofrecían los dispositivos analógicos tradicionales que a las interfaces que se utilizan en los sistemas informáticos, para configurar las magnitudes a adquirir y sus rangos y para mostrar los resultados.

2.2. Tipos de instrumentación.

Atendiendo al tipo de magnitudes medidas y a la forma de hacerlo, se pueden destacar las clases que a continuación se citan.

2.2.1. Analógica.

Es la instrumentación encargada de medir magnitudes analógicas, es decir, aquéllas que pueden variar de forma contínua de valor dentro de su rango. Los sistemas digitales que se utilizan dentro de esta clase deben ser capaces de convertir las magnitudes a señales eléctricas mediante el transductor correspondiente y de convertir estas señales a valores digitales mediante conversores analógico-digitales. El número de bits de la conversión y, en algunos casos, su velocidad son un indicativo de la calidad del aparato.

Hay una gran cantidad de instrumentos de este tipo. En la actualidad los más utilizados en el mantenimiento de sistemas informáticos serían los multímetros y los osciloscopios.

2.2.2. Digital.

Para muestrear señales digitales, propias de todos los sistemas informáticos, se utiliza instrumentación de este tipo, que simplemente compara una magnitud eléctrica –tensión o, más raramente, corriente– con un umbral para ver si representa un cero o un uno lógico. El valor del umbral y la polaridad de la comparación será acorde con los niveles eléctricos de la lógica que se esté utilizando. Por ejemplo el estándar RS232 utiliza lógica negativa señalizando el uno lógico con una tensión entre -3 y -12 V y el cero entre +3 y +12 V. Un analizador lógico configurado para medir señales de este tipo pondría su umbral en un valor arbitrario entre -3 y +3 –normalmente 0 V– y utilizaría polaridad invertida.

Como puede verse en el ejemplo anterior, el sistema digital no puede determinar si las señales están fuera del rango eléctrico aceptable –en el caso del ejemplo, las señales entre +3 y -3 V serían tratadas como valores lógicos correctos. Esto demuestra otra característica de la instrumentación digital: debe usarse para verificar los aspectos lógicos de los sistemas, no siendo adecuada para detectar problemas eléctricos –a diferencia del anterior tipo. Sin embargo, siguiendo con el ejemplo, un osciloscopio puede usarse, aunque de forma más incómoda, para verificar la validez lógica de un conjunto pequeño de señales digitales.

Dado que la adquisición de la señal es bastante más simple las características que definen la calidad de esta clase son la velocidad –que suele ser mucho mayor que la de los sistemas analógicos– y la profundidad de los canales de medida –es decir, el número de muestras que pueden almacenar.

El instrumento más utilizado de esta clase es el analizador lógico. Se usan también las sondas lógicas para verificación rápida de pocos niveles lógicos en puntos de fácil acceso.

2.2.3. Mixta.

De los párrafos anteriores se desprende que es bueno, para ciertos análisis, disponer de una gran cantidad de medidas lógicas y poder a la vez y de forma síncronizada acceder a los valores eléctricos de algunas de ellas. Para ello se utilizan los analizadores mixtos, que vienen a ser como la unión de un analizador lógico y un osciloscopio. Estos sistemas suelen construirse de forma modular, en bastidores que incluyen elementos de control, sincronizadores y proceso comunes, por medio de tarjetas con varios –64, 128...– canales digitales y otras con varios -2, 4...- canales analógicos.

Otros elementos de adquisición mixtos que se utilizan sobre todo en control y mucho menos en instrumentación son las tarjetas de adquisición para PC o los módulos de adquisición para autómatas. Son sistemas más o menos configurables con entradas analógicas y digitales. En este caso el número de entradas digitales suele ser mayor que el de analógicas, si bien la diferencia no es tan apreciable como entre los analizadores lógicos y los osciloscopios.

2.2.4. Comunicaciones.

Dentro de la instrumentación para instalaciones informáticas existe una clase de sistemas de medida de aplicación en el análisis de redes de comunicaciones que abarca el estudio anlógico del cableado y la transmisión de las señales, el estudio de la interpretación digital de las mismas y el análisis de protocolos a los distintos niveles de la arquitectura de la red. Por sus características especiales de sistema multinivel y por la especifidad de su campo de aplicación, es adecuado clasificarlos en una categoría propia.

2.3. Estándares.

En todos los ámbitos del desarrollo tecnológico con amplia aplicación industrial es una práctica recomendable la creación de estándares para homogeneizar los productos y desarrollos, aunar y, por lo tanto, ahorrar esfuerzos, permitir la interoperabilidad entre productos de diferentes fabricantes, etcétera. Esta tendencia se da también en el ámbito de los sistemas de medida e inspección, necesarios para el mantenimiento. Por ejemplo, en automoción existe el estándar FMS que establece el medio físico y el formato de la información que deben enviar las distintas centralitas de un vehículo y, de este modo, toda la información puede ser capturada y analizada por equipamiento estándar, válido para todos los vehículos que se adhieren al FMS.

En el ámbito del mantenimiento de equipos inforáticos existen también estos estándares. Los más importantes se van a presentar a continuación.

2.3.1. Estándares de test.

El estandar IEEE 1149.1 Boundary Scan, también conocido como JTAG –Joint Test Action Group– fue definido en 1990 y sirve para verificar el buen funcionamiento de los sistemas físicos –hardware–, así como para proveer de funcionalidad adicional a algunos circuitos digitales.

Los circuitos digitales que se adhieren al estándar añaden 4 pines más en su encapsulado, **TDI**, **TDO**, **TCK** y **TMS**, además de contar con una descripción del comportamiento del circuito empleada por el software de test que permite verificar si los datos obtenidos mediante el test son o no correctos. El funcionamiento a grandes rasgos del estándar es el que sigue:

Los sistemas se diseñan con las señales **TDI**, *Test Data Input* y **TDO**, *Test Data Ouput*, de todos los circuitos conectadas formando un anillo serie que recorre todos los dispositivos que cumplen el estándar. Las señales **TCK**, *Test Clock* y **TMS**, *Test Mode Select* se conectan en paralelo a todos estos circuitos. Desde un conector externo con las cuatro señales, en que **TDI** y **TDO** son, respectivamente, el origen y final del anillo serie, y mediante el hardware de conexión adecuado, el software inicia el test, recoge los datos del mismo y los compara con los resultados esperados según la descripción que posee de los circuitos.

El test es básicamente una ejecución real del funcionamiento del sistema, en que las entradas y salidas de todos los pines son capturadas y enviadas vía serie por las conexiones indicadas. De esta forma se pueden detectar problemas en las soldaduras, pistas del circuito impreso, etcétera.

Una descripción más detallada del estándar y su funcionamiento se puede encontrar en

`http://www.jtag.org/products/Boundary-Scan_Tutorial.htm`

2.3.2. Estándares de comunicación.

El estándar IEEE 488 GPIB *General Purpose Interface Bus* se basa en el HPIB *Hewlett Packard Interface Bus* aparecido en los años 60 y sirve para comunicar instrumentos y coordinar y centralizar su funcionamiento, es decir, para conectar instrumentos a ordenadores para realizar tests de forma conjunta y recoger y analizar los resultados en un sistema central. Es un bus de comunicaciones simple que permite enviar órdenes y datos entre los distintos agentes, para configurar dispositivos, realizar medidas y comunicar los resultados.

3. Herramientas lógicas (software).

Una buena parte del funcionamiento de los sistemas informáticos se debe a las aplicaciones, controladores y sistemas operativos, es decir, al software que se ejecuta sobre sus componentes físicos. Además el mal funcionamiento de algunos de estos componentes no siempre compromete totalmente el funcionamiento del sistema, sino que estos fallos se manifiestan como averías de algunas partes o funciones del sistema, mientras que los servicios básicos se siguen satisfaciendo de manera más o menos correcta.

Por este motivo, una buena cantidad de herramientas usadas para el mantenimiento de los sistemas informáticos son programas, herramientas software, que se ejecutan sobre el mismo sistema que están analizando.

3.1. Características.

Cuando el sistema objeto de las actuaciones de mantenimiento funciona parcialmente, las aplicaciones de mantenimiento son herramientas muy potentes porque permiten estudiar desde dentro del propio sistema cómo se comporta. Desde analizar en detalle los distintos componentes hasta mejorar las configuración del sistema, el software es una herramienta de una gran versatilidad y potencia a la hora de estudiar el sistema sobre el que se ejecuta.

En algunos casos, lógicamente, y dado que es el propio sistema analizado quien realiza los análisis, las herramientas software serán insuficientes pues enmascaran o sufren los propios fallos que se quiere eliminar.

Las aplicaciones que se utilizan para el mantenimiento de los sistemas informáticos son, generalmente, aplicaciones multinivel, es decir, que se actúan directamente sobre el hardware con mayor prioridad incluso que el sistema operativo mientras que por otra parte ofrecen la información acerca de los resultados de las pruebas mediante interfaces de usuario elaboradas. La versatilidad del software hace que estas aplicaciones puedan obtener información del propio hardware, del sistema operativo, de aplicaciones del sistema, etcétera y que se pueda configurar el medio de salida según los requisitos del caso, utilizando la interfaz de ventanas como se ha dicho antes, mediante salida simple en consola, ficheros de registro e incluso a través de conexiones de red o de correo electrónico.

El principal inconveniente de este tipo de herramientas es la intrusión que provocan en el sistema que están analizando. Como se ha dicho anteriormente, los instrumentos de medida ideales deben obtener datos del sistema sin modificarlo. En el caso del software, siempre que instalamos y ejecutamos una aplicación estamos modificando la configuración del equipo objeto de estudio. Si el problema que queremos detectar es de bajo nivel, funcionamiento del hardware, las herramientas software dan muy buen resultado pues no modifican en absoluto el hardware del equipo. Si, por el contrario, los problemas se dan en la interacción entre módulos software –principalmente en el sistema operativo o en la interacción entre el sistema y las aplicaciones– el añadir un proceso nuevo a la configuración que causa el error puede, con cierta probabilidad, enmascarar el problema. En este caso se debería probar con distintas aplicaciones de análisis –o con distintas configuraciones del sistema– hasta que la avería se reproduzca de nuevo.

La principal ventaja del software es su versatilidad y flexibilidad. Al hablar de aplicaciones para el mantenimiento de equipos informáticos, esta versatilidad ofrece una potencia elevada a un coste muy reducido. Como ya se ha comentado, las aplicaciones pueden ser configuradas para presentar los datos de muy distinta forma: gráficamente, como texto, en ficheros, por red, etcétera. Además se pueden configurar para realizar tests de forma local y presencial o remota y temporizada, incluso se pueden realizar tests periódicamente enviando posteriormente los resultados a una máquina de administración o directamente al administrador por correo electrónico.

La posibilidad de utilizar las conexiones de red permite también ejecutar remotamente las aplicaciones, crear aplicaciones de test distribuidas para verificar el funcionamiento de la red de interconexión, etcétera.

Resumiendo, el software de análisis de sistemas informáticos es una herramienta de potencia ilimitada que permite la mayor flexibilidad y configurabilidad a la vez que aporta por lo general muy buenos resultados en el mantenimiento de los ordenadores.

3.2. Tipos.

Una vez comentadas las características del software como herramienta para el mantenimiento se va a presentar una clasificación del mismo según su aplicación, destacando los aspectos y usos más importantes en cada uno de los casos.

3.2.1. Depuración de hardware.

En este caso se trata de aplicaciones que se instalan en un equipo y permiten acceder al hardware del sistema, verficar su funcionamiento y su interacción con el sistema operativo y los manejadores. Generalmente son aplicaciones que dan unos resultados muy buenos pues no interactúan de forma negativa con otros componentes software del equipo en el que actúan.

Se pueden distinguir los siguientes grupos:

- Detección y verificación del hardware. Estas aplicaciones acceden directamente a los dispositivos del ordenador, por lo tanto se instalan o ejecutan por debajo del sistema operativo. Permiten identificar el hardware presente en el sistema, sus versiones, recursos, etcétera y realizar diversas pruebas sobre los dispositivos físicos. Son herramientas muy útiles para la verificación de la memoria, los recursos de almacenamiento y el propio procesador.

- Ayudas a la configuración. Son herramientas que pueden instalarse por encima del sistema operativo, aunque con permisos de acceso al hardware. Ayudan a identificar los componentes del sistema y a verificar su acceso directo y a través de los manejadores del sistema operativo. De esta forma ayudan a determinar la configuración adecuada del software del sistema.

- Código empotrado en el sistema. Cada vez proliferan más las aplicaciones que se instalan como partes del sistema operativo y que permiten verificar y obtener información acerca del funcionamiento del hardware. Existen monitores de temperatura y tensión del procesador, de estado de los discos duros utilizando el estándar SMART, de utilización de la memoria, de los recursos del sistema, etcétera. Se bueno tener este tipo de monitores instalados en cualquier sistema del que se quiera hacer un mantenimiento preventivo adecuado y cómodo.

3.2.2. Depuración de software.

A la hora de hablar de programas para la depuración del software se debe comentar un hecho fundamental, derivado de su propia naturaleza: el software no sufre desgaste, no se deteriora dado que no tiene existencia física real. Así pues, si hablamos de mantenimiento del software debemos referirnos a una de las siguientes circunstancias particulares:

- aplicación de actualizaciones y parches de mantenimiento del sistema operativo o aplicaciones. Esta es una tarea de administración del sistema, de ejecución previsible y que no requiere propiamente de herramientas de mantenimiento para llevarse a cabo.

- Diagnóstico y reparación –en su caso– de problemas en las aplicaciones, en el sistema operativo o en la interacción entre las aplicaciones y el sistema. En este caso sí se puede hablar propiamente de mantenimiento, debido a que las aplicaciones, el sistema operativo o ambos, no están totalmente libres de fallos y estos se manifiestan de forma esporádica. Aunque algunas de las herramientas que se explicarán a continuación se pueden utilizar en este caso, su uso debería limitarse al diagnóstico del problema. Una vez encontrado, será responsabilidad de los suministradores de la aplicación o del sistema operativo el subsanarlos. La única excepción a esa regla se da cuando el problema sea debido a falta de algún recurso del sistema –por ejemplo, espacio en disco– o a una mala configuración del sistema o la aplicación, caso en el que sí se podrá llegar a una solución local.

- Herramientas para la depuración y diagnóstico de aplicaciones a disposición de los propios programadores, para eliminar el mayor número de fallos en la aplicación antes de su explotación y para ayudar a solucionar

los problemas del apartado anterior durante el mantenimiento del sistema informático. Las herramientas que se describen a continuación son válidas fundamentalmente para estas actuaciones.

A continuación se describen las herramientas de que se dispone para hacer aplicaciones fácilmente mantenibles y que en algunos casos permiten detectar e incluso solucionar fallos durante su explotación.

- Depuradores –*debuggers*– y *profilers*. Los depuradores de código fuente permiten ejecutar el programa compilado o interpretado y observar el estado –variables y memoria– de su entorno durante la ejecución, deteniéndola mediante puntos de ruptura o actualizando en tiempo casi real las variables inspeccionadas. Los puntos de ruptura se pueden disparar incondicionalmente o estableciendo condiciones de datos, iteraciones, etcétera. De esta manera se puede revisar el código y eliminar errores. Una vez el programa se considera validado se compila –en su caso– eliminando la información de depuración –la que relaciona el código compilado con el fuente– para obtener la versión de distribución.

  Los *profilers* permiten ejecutar el programa una serie de veces con datos de entrada sintéticos –vectores de prueba– analizando el tiempo que se consume en la ejecución de cada una de las funciones, bucles y construcciones más importantes del código. De esta manera se puede, en una segunda pasada, refinar aquellos bloques de código que consumen más tiempo de ejecución para hacerlos más eficientes y rápidos, mejorando sustancialmente el tiempo de ejecución total.

- Depuradores –*debuggers*– y desensambladores. Cualquier programa ejecutable, se disponga o no de su código fuente, es susceptible de ser depurado con un depurador estándar del sistema y su código analizado con un desensamblador. Esta práctica, sin embargo, es más bien propia del *hacking* que del mantenimiento de las aplicaciones, aunque en algún caso esporádico pueden ayudar a diagnosticar algún problema de interacción entre la aplicación y el sistema.

  Los depuradores de ejecutables son similares a los de código fuente pero permiten exclusivamente la inspección de los registros de la arquitectura y de posiciones de memoria, no de las variables propiamente dichas, cuyos nombres y tipos son desconocidos.

- Código de depuración en las aplicaciones y buenas prácticas de programación. Sin lugar a dudas, la mejor forma de hacer aplicaciones fáciles de depurar y de mantener consiste en tener estos dos fines en mente durante el desarrollo de toda la aplicación. De esta forma, incluso sin la ayuda de un depurador se pueden solucionar la mayor parte de los errores y progresar de manera más fácil y rápida hacia nuevas versiones de la misma.

  Durante el desarrollo de los programas es común poner nuestros propios inspectores de ciertas variables mediante la salida de sus valores por pantalla durante la ejecución de los programas. Este código se elimina una vez se ha comprobado el buen funcionamiento de bloque que se está depurando. Una forma adecuada de llevar a cabo este proceso es incluir el código de depuración dentro de bloques de compilación condicional, que se incluirán o no en el código compilado en función de algún parámetro.

  En lenguaje C, por ejemplo, se podría tener

```
#define DEBUG
...
#ifdef DEBUG
printf("El valor del area en la iteracion %d es: %f\n", i, area);
#endif
```

  donde la sentencia `printf` se ejecutaría sólo si tenemos el parámetro `DEBUG` definido. Comentando la primera línea, el código que se compila condicionalmente se eliminaría totalmente del binario compilado.

  El entorno estándar del lenguaje C nos ofrece otra forma de incluir sentencias de depuración condicionalmente mediante la función `void assert(int expresion)`, que se encuentra definida en `assert.h`. Normalmente `expresion` es una comparación, que si se evalúa como falsa, es decir, con su valor igual a cero, durante la ejecución, fuerza que el programa aborte indicando, mediante un mensaje por pantalla, en qué fichero, función y línea se encuentra la llamada a `assert` que produjo el fallo. El funcionamiento descrito se da si la constante `NDEBUG` no se encuentra definida. En caso contrario, si definimos `NDEBUG`, la llamada a `assert()` no genera ningún código ni produce ningún efecto. Un funcionamiento similar, pero

usando el valor de un código de error del sistema, tiene la función `assert_perror()`. Se puede ampliar la información sobre estas funciones y su uso en las páginas de manual de cualquier sistema linux –mediante las órdenes `man assert` y `man assert_perror`.

También es importante que las aplicaciones desarrolladas sean capaces de detectar en la medida de lo posible los errores que se puedan producir y de informar acerca de ellos para solucionarlos –si se trata de un error del sistema que afecta a la aplicación– o para solicitar su corrección si se trata de un error en la propia aplicación. Para esto es necesario en primer lugar seguir una práctica de programación que utilice al máximo toda la información disponible en tiempo de ejecución.

Todas las llamadas al sistema y la mayor parte de las funciones de las bibliotecas estándar devuelven codificada en el valor devuelto –usualmente devolviendo un valor menor que cero– información acerca de si se ha producido un error en la llamada. En este caso, una variable del sistema guarda un código que identifica el error producido. En lenguaje C esta variable es `errno`, y existe además la función `perror(char *s)` que envía a la salida de error estándar el texto que le pasamos como argumento y otro –definido en el vector `sys_errlist[]` e indexado por el valor de `errno`– que indica el error producido –de nuevo, se puede ampliar esta información en las páginas de manual. Junto a la práctica de verificar el valor devuelto por todas las llamadas al sistema o a las bibliotecas, es conveniente incluir un comportamiento similar en nuestras propias funciones, de manera que devuelvan un indicador de error si se da alguna circunstancia anómala durante su ejecución.

En general, es bueno que una aplicación detecte todas las posibles circunstancias erróneas y que informe sobre ellas, abortando o no su ejecución según lo críticas que sean. También es bueno a veces que las aplicaciones informen de lo que va ocurriendo durante su ejecución, sea anómalo u ordinario, sobre todo si las aplicaciones no se comunican directamente con el usuario –demonios o servicios del sistema, servicios de red, aplicaciones intermedias, etcétera. Sin embargo, cuando todo funciona correctamente, el exceso de información puede ser molesto para el usuario o para el propio sistema. Por todo esto, otra práctica común en la programación y que debería adoptarse en toda aplicación en aras de su mantenibilidad, es la de definir diferentes niveles de información –`verbose levels` en inglés– configurables a la hora de ejecutarla –usualmente con el parámetro `-v N`, donde `N` indica el nivel. Así en el nivel más bajo se informará sólo de los errores, y en el más alto de todo aquéllo que se va produciendo durante la ejecución y puede ser de utilizad para verificar lo que está ocurriendo durante el proceso. En este caso, el código que genera los mensajes se encuentra necesariamente incorporado en el ejecutable y se ejecuta o no mediante selección condicional en tiempo de ejecución.

Los mensajes que se generan desde una aplicación pueden dirigirse a la salida estándar o, más adecuadamente, a un fichero de registro o `log` cuyo nombre y, al menos, ubicación, deberían ser configurables.

3.2.3. Aplicaciones mixtas.

Existen para ciertas aplicaciones especiales instrumentos formados por hardware que se inserta en el sistema a medir y software que se ejecuta sobre él. Un ejemplo muy específico es el de ciertas tarjetas para depuración de hardware–software en PCs, que llevan además software de test propio.

El ejemplo más claro de este tipo de sistemas mixtos es en el análisis de redes. La unión de software ejecutándose en los nodos de la red y hardware instalado en ella permite identificar problemas de conectividad y protocolo en los ordenadores, cableado y equipamiento intermedio de la red.

3.2.4. Otros.

Además del uso del software como herramienta de diagnóstico y detección de problemas, su uso como herramienta auxiliar es muy difundido y extenso. A continuación se van a citar algunos ejemplos de este uso, aunque la lista podría ser mucho mayor.

- Software de conexión remota para instalar, verificar o actuar sobre un equipo que se está manteniendo sin necesidad de desplazarse a su ubicación.
- Máquinas virtuales que permiten depurar sistemas opertivos o crear redes y verificar el comportamiento de las aplicaciones en red sin necesidad de utilizar más de un equípo físico.
- Software de envío de avisos, alarmas, registros o cualquier información relevante de forma remota por correo electronico.

- Software de temporización y realización programada de actividades de mantenimiento como copias de seguridad, desfragmentación, etcétera.
- ...

4. Herramientas para la gestión del mantenimiento.

Las aplicaciones software para la gestión están implantadas con resultados exitosos desde hace mucho tiempo en todos los procesos y negocios en que se pueda pensar. Uno de estos procesos es el de mantenimiento en sí, por lo tanto es normal la existencia de aplicaciones para la gestión del mantenimiento de las empresas. Estas aplicaciones específicas gestionan todos los aspectos del mantenimiento, en particular

- Procedimientos de mantenimiento preventivo y correctivo y su planificación.
- Stocks de materiales para el mantenimiento, consumibles e inventariables.
- Equipos disponibles, periodos de garantía y gestión de la obsolescencia.
- Costes del mantenimiento.
- ...