

reemplazar el escritorio estándar de la GUI (*Windows Explorer*) con un programa distinto, para lo cual se modifican ciertos valores en el registro, aunque pocas personas hacen esto.

1.5.7 La ontogenia recapitula la filogenia

Después de que se publicó el libro de Charles Darwin titulado *El origen de las especies*, el zoólogo alemán Ernst Haeckel declaró que “la ontogenia recapitula la filogenia”. Lo que quiso decir fue que el desarrollo de un embrión (ontogenia) repite (es decir, recapitula) la evolución de las especies (filogenia). En otras palabras, después de la fertilización un óvulo humano pasa a través de las etapas de ser un pez, un cerdo y así en lo sucesivo, hasta convertirse en un bebé humano. Los biólogos modernos consideran esto como una simplificación burda, pero aún así tiene cierto grado de verdad.

Algo análogo ha ocurrido en la industria de las computadoras. Cada nueva especie (mainframe, minicomputadora, computadora personal, computadora de bolsillo, computadora de sistema integrado, tarjeta inteligente, etc.) parece pasar a través del desarrollo que hicieron sus ancestros, tanto en hardware como en software. A menudo olvidamos que la mayor parte de lo que ocurre en el negocio de las computadoras y en muchos otros campos está controlado por la tecnología. La razón por la que los antiguos romanos no tenían autos no es que les gustara caminar mucho; se debió a que no sabían construirlos. Las computadoras personales existen *no* debido a que millones de personas tienen un deseo reprimido durante siglos de poseer una computadora, sino a que ahora es posible fabricarlas a un costo económico. A menudo olvidamos qué tanto afecta la tecnología a nuestra visión de los sistemas y vale la pena reflexionar sobre este punto de vez en cuando.

En especial, con frecuencia ocurre que un cambio en la tecnología hace que una idea se vuelva obsoleta y desaparece con rapidez. Sin embargo, otro cambio en la tecnología podría revivirla de nuevo. Esto es en especial verdadero cuando el cambio tiene que ver con el rendimiento relativo de distintas partes del sistema. Por ejemplo, cuando las CPUs se volvieron mucho más rápidas que las memorias, las cachés tomaron importancia para agilizar la memoria “lenta”. Si algún día la nueva tecnología de memoria hace que las memorias sean mucho más rápidas que las CPUs, las cachés desaparecerán. Y si una nueva tecnología de CPUs las hace más rápidas que las memorias de nuevo, las cachés volverán a aparecer. En biología, la extinción es para siempre, pero en la ciencia computacional, algunas veces sólo es durante unos cuantos años.

Como consecuencia de esta transitoriedad, en este libro analizaremos de vez en cuando conceptos “obsoletos”, es decir, ideas que no son óptimas con la tecnología actual. Sin embargo, los cambios en la tecnología pueden traer de nuevo algunos de los denominados “conceptos obsoletos”. Por esta razón, es importante comprender por qué un concepto es obsoleto y qué cambios en el entorno pueden hacer que vuelva de nuevo.

Para aclarar aún más este punto consideremos un ejemplo simple. Las primeras computadoras tenían conjuntos de instrucciones cableados de forma fija. Las instrucciones se ejecutaban directamente por el hardware y no se podían modificar. Después llegó la microprogramación (primero se introdujo en gran escala con la IBM 360), en la que un intérprete subyacente ejecutaba las “instrucciones de hardware” en el software. La ejecución de instrucciones fijas se volvió obsoleta. Pero esto no era lo bastante flexible. Después se inventaron las computadoras RISC y la microprogramación

(es decir, la ejecución interpretada) se volvió obsoleta, debido a que la ejecución directa era más veloz. Ahora estamos viendo el resurgimiento de la interpretación en forma de applets de Java que se envían a través de Internet y se interpretan al momento de su llegada. La velocidad de ejecución no siempre es crucial, debido a que los retrasos en la red son tan grandes que tienden a dominar. Por ende, el péndulo ha oscilado varias veces entre la ejecución directa y la interpretación y puede volver a oscilar de nuevo en el futuro.

Memorias extensas

Ahora vamos a examinar algunos desarrollos históricos en el hardware y la forma en que han afectado al software repetidas veces. Las primeras mainframes tenían memoria limitada. Una IBM 7090 o 7094 completamente equipada, que fungió como rey de la montaña desde finales de 1959 hasta 1964, tenía cerca de 128 KB de memoria. En su mayor parte se programaba en lenguaje ensamblador y su sistema operativo estaba escrito en lenguaje ensamblador también para ahorrar la valiosa memoria.

A medida que pasaba el tiempo, los compiladores para lenguajes como FORTRAN y COBOL se hicieron lo bastante buenos como para que el lenguaje ensamblador se hiciera obsoleto. Pero cuando se liberó al mercado la primera minicomputadora comercial (PDP-1), sólo tenía 4096 palabras de 18 bits de memoria y el lenguaje ensamblador tuvo un regreso sorpresivo. Con el tiempo, las microcomputadoras adquirieron más memoria y los lenguajes de alto nivel prevalecieron.

Cuando las microcomputadoras llegaron a principios de 1980, las primeras tenían memorias de 4 KB y la programación en lenguaje ensamblador surgió de entre los muertos. A menudo, las computadoras embebidas utilizaban los mismos chips de CPU que las microcomputadoras (8080, Z80 y posteriormente 8086) y también se programaban en ensamblador al principio. Ahora sus descendientes, las computadoras personales, tienen mucha memoria y se programan en C, C++ y Java, además de otros lenguajes de alto nivel. Las tarjetas inteligentes están pasando por un desarrollo similar, aunque más allá de un cierto tamaño, a menudo tienen un intérprete de Java y ejecutan los programas de Java en forma interpretativa, en vez de que se compile Java al lenguaje máquina de la tarjeta inteligente.

Hardware de protección

Las primeras mainframes (como la IBM 7090/7094) no tenían hardware de protección, por lo que sólo ejecutaban un programa a la vez. Un programa con muchos errores podía acabar con el sistema operativo y hacer que la máquina fallara con facilidad. Con la introducción de la IBM 360, se hizo disponible una forma primitiva de protección de hardware y estas máquinas podían de esta forma contener varios programas en memoria al mismo tiempo, y dejarlos que tomaran turnos para ejecutarse (multiprogramación). La monoprogramación se declaró obsoleta.

Por lo menos hasta que apareció la primera minicomputadora (sin hardware de protección) la multiprogramación no fue posible. Aunque la PDP-1 y la PDP-8 no tenían hardware de protección, en cierto momento se agregó a la PDP-11 dando entrada a la multiprogramación y con el tiempo a UNIX.

Cuando se construyeron las primeras microcomputadoras, utilizaban el chip de CPU 8080 de Intel, que no tenía protección de hardware, por lo que regresamos de vuelta a la monoprogramación. No fue sino hasta el Intel 80286 que se agregó hardware de protección y se hizo posible la multiprogramación. Hasta la fecha, muchos sistemas integrados no tienen hardware de protección y ejecutan un solo programa.

Ahora veamos los sistemas operativos. Al principio, las primeras mainframes no tenían hardware de protección ni soporte para la multiprogramación, por lo que ejecutaban sistemas operativos simples que se encargaban de un programa cargado en forma manual a la vez. Más adelante adquirieron el soporte de hardware y del sistema operativo para manejar varios programas a la vez, después capacidades completas de tiempo compartido.

Cuando aparecieron las minicomputadoras por primera vez, tampoco tenían hardware de protección y ejecutaban un programa cargado en forma manual a la vez, aun cuando la multiprogramación estaba bien establecida en el mundo de las mainframes para ese entonces. Gradualmente adquirieron hardware de protección y la habilidad de ejecutar dos o más programas a la vez. Las primeras microcomputadoras también fueron capaces de ejecutar sólo un programa a la vez, pero más adelante adquirieron la habilidad de la multiprogramación. Las computadoras de bolsillo y las tarjetas inteligentes se fueron por la misma ruta.

En todos los casos, el desarrollo de software se rigió en base a la tecnología. Por ejemplo, las primeras microcomputadoras tenían cerca de 4 KB de memoria y no tenían hardware de protección. Los lenguajes de alto nivel y la multiprogramación eran demasiado para que un sistema tan pequeño pudiera hacerse cargo. A medida que las microcomputadoras evolucionaron en las computadoras personales modernas, adquirieron el hardware necesario y después el software necesario para manejar características más avanzadas. Es probable que este desarrollo continúe por varios años más. Otros campos también pueden tener esta rueda de reencarnaciones, pero en la industria de las computadoras parece girar con más velocidad.

Discos

Las primeras mainframes estaban en su mayor parte basadas en cinta magnética. Leían un programa de la cinta, lo compilaban, lo ejecutaban y escribían los resultados de vuelta en otra cinta. No había discos ni un concepto sobre el sistema de archivos. Eso empezó a cambiar cuando IBM introdujo el primer disco duro: el RAMAC (*RAndom Access*, Acceso aleatorio) en 1956. Ocupaba cerca de 4 metros cuadrados de espacio de piso y podía almacenar 5 millones de caracteres de 7 bits, lo suficiente como para una fotografía digital de mediana resolución. Pero con una renta anual de 35,000 dólares, ensamblar suficientes discos como para poder almacenar el equivalente de un rollo de película era en extremo costoso. Con el tiempo los precios disminuyeron y se desarrollaron los sistemas de archivos primitivos.

Uno de los nuevos desarrollos típicos de esa época fue la CDC 6600, introducida en 1964 y considerada durante años como la computadora más rápida del mundo. Los usuarios podían crear “archivos permanentes” al darles un nombre y esperar que ningún otro usuario hubiera decidido también que, por decir, “datos” fuera un nombre adecuado para un archivo. Éste era un directorio de un solo nivel. Con el tiempo las mainframes desarrollaron sistemas de archivos jerárquicos complejos, lo cual probablemente culminó con el sistema de archivos MULTICS.

Cuando las minicomputadoras empezaron a usarse, con el tiempo también tuvieron discos duros. El disco estándar en la PDP-11 cuando se introdujo en 1970 era el disco RK05, con una capacidad de 2.5 MB, aproximadamente la mitad del RAMAC de IBM, pero sólo tenía cerca de 40 cm de diámetro y 5 cm de altura. Pero también tenía al principio un directorio de un solo nivel. Cuando llegaron las microcomputadoras, CP/M fue al principio el sistema operativo dominante y también soportaba un solo directorio en el disco (flexible).

Memoria virtual

La memoria virtual (que se describe en el capítulo 3) proporciona la habilidad de ejecutar programas más extensos que la memoria física de la computadora, llevando y trayendo pedazos entre la RAM y el disco. Pasó por un desarrollo similar, ya que apareció primero en las mainframes, después avanzó a las minis y a las micros. La memoria virtual también permitió la capacidad de ligar dinámicamente un programa a una biblioteca en tiempo de ejecución, en vez de compilarlo. MULTICS fue el primer sistema operativo en tener esta capacidad. Con el tiempo, la idea se propagó descendiendo por toda la línea y ahora se utiliza ampliamente en la mayoría de los sistemas UNIX y Windows.

En todos estos desarrollos vemos ideas que se inventaron en un contexto y más adelante se descartaron cuando cambió el contexto (la programación en lenguaje ensamblador, la monoprogramación, los directorios de un solo nivel, etc.) sólo para reaparecer en un contexto distinto, a menudo una década más tarde. Por esta razón, en este libro algunas veces vemos ideas y algoritmos que pueden parecer atrasados en comparación con las PC de hoy en día con capacidades de gigabytes, pero que pronto pueden volver en las computadoras incrustadas y las tarjetas inteligentes.

1.6 LLAMADAS AL SISTEMA

Hemos visto que los sistemas operativos tienen dos funciones principales: proveer abstracciones a los programas de usuario y administrar los recursos de la computadora. En su mayor parte, la interacción entre los programas de usuario y el sistema operativo se relaciona con la primera función: por ejemplo, crear, escribir, leer y eliminar archivos. La parte de la administración de los recursos es en gran parte transparente para los usuarios y se realiza de manera automática. Por ende, la interfaz entre los programas de usuario y el sistema operativo trata principalmente acerca de cómo lidiar con las abstracciones. Para comprender realmente qué hacen los sistemas operativos, debemos examinar esta interfaz con detalle. Las llamadas al sistema disponibles en la interfaz varían de un sistema operativo a otro (aunque los conceptos subyacentes tienden a ser similares).

Por lo tanto, nos vemos obligados a elegir una opción entre 1) generalidades imprecisas (“los sistemas operativos tienen llamadas al sistema para leer archivos”) y 2) cierto sistema específico (“UNIX tiene una llamada al sistema conocida como `read` con tres parámetros: uno para especificar el archivo, uno para decir en dónde se van a colocar los datos y uno para indicar cuantos bytes se deben leer”).

Hemos optado por la última opción; implica más trabajo, pero proporciona una visión más detallada en cuanto a lo que realmente hacen los sistemas operativos. Aunque este análisis se refiere