Joaquín Ríos Boutín
(Laboratorio de Cálculo-
Facultad de Informática de
Barcelona-
U.P.C.)

UNIX = Sistema Operativo + Herramientas de Software

1. Introducción

Aunque el tema monográfico de este número de Novática está dedicado a los Entornos de Programación, UNIX no fue creado como tal, ni es exclusivamente un «software» de esta clase.

UNIX es un Sistema Operativo potenciado con un conjunto de herramientas de «software» que son las que lo incorporan a los Entornos de Programación.

Si se clasifican los Entornos de Programación en:

- Entornos concebidos como repertorio de herramientas
- Entornos concebidos como soporte de un método.

UNIX está incluído en el primer grupo [1]. Este Sistema Operativo fue creado en la década de los 70, pudiendo considerarle el más extendido y uno de los Entornos de Programación más aceptado [2], debido principalmente a su éxito como Sistema Operativo más que a la bondad y abundancia de las herramientas de «software» que contiene.

Puesto que en Novática no ha existido ninguna presentación de este «software», se ha dividido este artículo en las siguientes partes:

- Génesis del Sistema Operativo.
- Introducción al UNIX.
- El UNIX como Entorno de Programación.
- Por qué y para qué el Sistema Operativo Unix.
- Conclusiones.

2. Génesis del Sistema Operativo

UNIX es la recopilación de experiencias de Laboratorios Bell en los años 1965 a 1969 conjuntamente con General Electric, en el Massachusetts Institute of Technology, durante el desarrollo del sistema MULTICS, siendo construído este Sistema Operativo para ser utilizado en grandes computadores [3].

Los Laboratorios Bell pensaron en UNIX como un Sistema Operativo para sus miniordenadores de 16 bits, aportando las ventajas y nuevas experiencias adquiridas en el proyecto anterior. Además, dadas las necesidades de la compañía, debía de estar orientado a facilitar al máximo la programación, ya fuera de investigación como de desarrollo, por lo que debía de ser un Entorno de Programación orientando a determinadas características de investigación y desarrollo.

La primera versión de UNIX aparece al principio de la década de los 70, escrita en ensamblador y funcionando en un DEC PDP7.

En 1972 se escribe la primera versión de UNIX en lenguaje C, lenguaje de sistemas de alto nivel, lo que conlleva la posibilidad de transportabilidad y una mejor comprensión del mismo. Además, en esta versión se introduce la multiprogramación y la posibilidad de compartir código reentrante.

Seguidamente, dada la difusión de la familia de miniordenadores PDP 11 en las universidades y en la misma empresa, Laboratorios Bell hace una versión de UNIX operativa para éstos.

En 1973 la Western Electric, empresa del grupo Bell, concede licencias de utilización a numerosas universidades americanas, creándose en 1975 el primer grupo de usuarios del UNIX, USENIX.

Es también en 1973 cuando aparece la versión 6 de UNIX que es la primera de más extensión, y en 1979 la versión 7 es ya un producto comercial.

A finales de la década de los 70 la Universidad de Berkeley ofrece su versión de UNIX que se caracteriza por poseer una gran librería de herramientas de «software».

En este entorno, en 1980, Laboratorios Bell con el propósito de desligarse de la imagen de monopolio del UNIX y, por otro lado, dada la gran reputación de Microsoft en el campo de desarrollo de «software» para microordenadores, se anuncia la aparición del XENIX, como una versión revisada y ampliada del UNIX que distribuirá Microsoft para microordenadores de 16 bits.

En la actualidad el número de fuentes de UNIX se ha incrementado de forma exponencial, ya sea por oferta directa del Sistema Operativo, ya sea por estar incorporado en los ordenadores de empresas informáticas que poseen licencias de explotación, dando lugar a una serie interminable de nombres como IDRIS, CROMIX, ZEUS, COHERENT, EUNIS, HPUX, ...[4].

Por otro lado, es también elevado el número de ordenadores que soportan este Sistema Operativo, desde los microordenadores de 16 bits basados en los microprocesadores Z8000, M68000, Intel 8086 o NS 16032, hasta los «mainframes» tipo IBM 43XX, pasando por los super o megaminiordenadores como VAX, GOULD-SEL 32XX, DG X000 o PE 32XX. La diversidad de potencia de estos sistemas hace que UNIX soporte un número máximo de terminales y de capacidades de disco distinto en cada uno de ellos.

Como consecuencia, la pluralidad de utilización de este Sistema Operativo hace entrever una clara intencionalidad de estandarización y unas posibilidades enormes de transportabilidad de «software» y de formación del usuario.

3. Introducción al UNIX

UNIX es un sistema Operativo de tiempo compartido y multiprogramación. Inicialmente estaba pensado para pequeños Centros de Cálculo con un número entre 10 y 20 terminales.

El Sistema Operativo UNIX está construído básica-

mente en dos capas [5]. La capa inferior es el núcleo o «kernal» y la capa superior son los procesos que están soportados por el núcleo. Los usuarios de UNIX son considerados como una capa externa a la capa de procesos y los discos junto con el resto de periféricos una capa interior al núcleo [6]. En la figura 1 se muestra de forma esquemática la organización de UNIX, mostrándose en ella las siguientes zonas:

- Una capa interactiva constituída por los usuarios tecleando en los terminales. Los comandos tecleados son enunciados al procesador de comandos del UNIX denominado «shell» (caparazón) que recibe este nombre porque desde el punto de vista del usuario envuelve el resto de elementos del UNIX y desde el punto de vista del Sistema Operativo rodea a los usuarios para comunicar con ellos. Hay que señalar que existen otros programas que también aceptan comandos y comunican directamente con el núcleo, como es el caso del editor de UNIX.
- Los Procesos que comunican con el usuario y acceden al núcleo. Los programas como el «shell» y el editor se ejecutan como procesos. Algunos de ellos realizan funciones de sistema como crear directorios de ficheros o listar ficheros por impresora.

Cada proceso tiene su propia memoria virtual o espacio de direcciones, de forma que no produce un solapamiento entre procesos o con el núcleo. La única forma que un proceso tiene para comunicar con otro proceso o con el núcleo es mediante una llamada al núcleo («trap»). Según el tipo de gestión de memoria, el núcleo hace «swapping» o pagina los procesos al disco.

- El Núcleo que soporta los procesos y el sistema de ficheros residentes en disco, atendiendo también las llamadas de los procesos al sistema y gestionando las interrupciones y control de periféricos.
- Los discos y el resto de periféricos. El disco contiene las estructuras de datos que son los ficheros de los usuarios, estando también controlado por el núcleo el acceso al resto de periféricos que hace que se comporten como unos «ficheros especiales» dentro del subdirectorio/dev [6], siendo interesante resaltar que en este subdirectorio está incluída la memoria del usuario y la memoria del núcleo.

console	– la consola del sistema
fd0	– floppy 0
fd1	– floppy 1
hd0	– disco duro 0
hd1	– disco duro 1
kmem	– memoria del núcleo
lp	– impresora
mem	– memoria del usuario
null	– periférico nulo (saco de bits)
phone	– marcador telefónico (a veces «acu»)
rfd0	– acceso de bajo nivel al floppy 0
rfd1	– acceso de bajo nivel al floppy 1
rhd0	– acceso de bajo nivel al disco duro 0
rhd1	– acceso de bajo nivel al disco duro 1
tty 0	– «teletipo» 0 – un terminal o modem
tty 1	– «teletipo» 1 – un terminal o modem
tty 2	– «teletipo» 2 – un terminal o modem
tty 3	– «teletipo» 3 – un terminal o modem

Tabla 1.– Los «ficheros especiales» del subdirectorio/dev

Seguidamente se detallan aquellas características que en el momento de aparición de UNIX lo convirtieron en una herramienta muy útil para el desarrollo de «software», las cuales, en gran parte, han sido incorporadas a los Sistemas Operativos actuales.

– El Sistema de ficheros:

La frase con que se puede definir la filosofía de concepción es: «el fichero contiene cualquier dato que el usuario ha decidido ponerle y no tiene más estructura que la

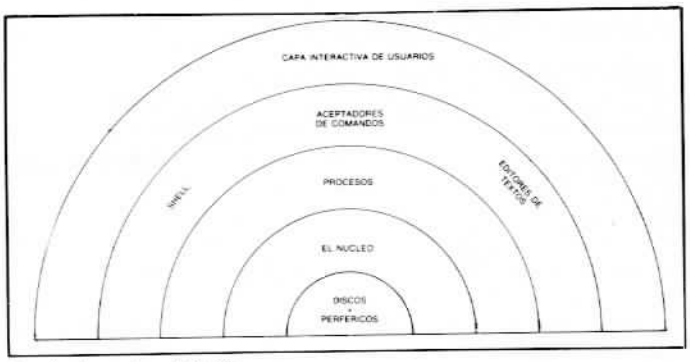

Fig. 1. La esfera UNIX.

impuesta por el propio usuario», lo que quiere decir que el Sistema de ficheros del UNIX es un conjunto de caracteres direccionables al azar, siendo el tamaño de un fichero exactamente el del número de caracteres que contiene, pudiendo tener como máximo un billón.

El sistema de emplear una cadena de caracteres en vez de un sistema basado en registros tiene la ventaja de hacer fácil la independencia del periférico al que se accede y la transparencia entre ficheros, periféricos y el resto de elementos de UNIX integrados en el Sistema de ficheros, formando parte de los «ficheros especiales» que son, ni más ni menos, que otros periféricos.

El construir métodos orientados a registros de longitud fija o variable, basados en una secuencia de caracteres, no es complejo y se deja como responsabilidad del usuario, si bien esto último hace perder algo de rendimiento del sistema en algunas aplicaciones comerciales, aspecto que se comentará posteriormente en la evaluación del UNIX.

Como consecuencia inmediata de esta sencillez aparece el hecho de que el sistema de Entrada/Salida está únicamente manejado por cinco llamadas básicas que son:

1. «open» (abrir fichero)
   df = open (nombrefich, modo)

en la que «modo» indica si se trata de sólo lectura, escritura, o ambas, siendo «df» el descriptor del fichero que se empleará en todas las referencias siguientes al fichero abierto.

2. «create» y «unlink» (crear y borrar un fichero)
   df = create (nombrefich, modo)
   df = unlink (nombrefich)
3. «read» y «write» (leer y escribir una cadena de caracteres)
   num_bytes_leídos = read (df, buffer, num_bytes_por_leer)
   num_bytes_escritos = write (df, buffer, num_bytes_por_escribir)
4. «seek» (comando de posicionado)
   seek (df, desplazamiento, tipo_desplazamiento)

en la que el «tipo_de_desplazamiento» puede ser relativo o absoluto, por bytes o en bloques de 512 que es el tamaño de los bloques lógicos de UNIX.

5. «close» (cerrar fichero)
   close (df)

Hay que hacer notar que como todos los periféricos están en la estructura de directorio, se puede utilizar «seek» en una cinta magnética, pero la eficiencia en tal caso sería muy baja.

Estas cinco llamadas básicas se complementan con algunos comandos más de entrada/salida como «gtty», «stty» y «stat» que se emplean para obtener información y dar mandatos a los terminales y ficheros.

La estructura de directorio es un árbol, lo que quiere decir que cada directorio posee punteros a ficheros normales y a otros ficheros que son a su vez directorios de otros ficheros, repitiéndose esta organización de forma

recursiva. Así, el directorio que no tiene puntero de ningún otro directorio se denomina directorio raíz, identificándolo con el símbolo «/», basándose en él toda la estructura de subdirectorios. En la figura 2 se muestra una estructura típica de subdirectorios del UNIX, siendo los elementos enmarcados directorios y el resto ficheros convencionales de programas o datos [6]. Hay que hacer notar que todo esto permite el empleo de nombres de ficheros iguales bajo subdirectorios diferentes.

Cabe señalar que desde un punto de vista de entorno de programación ésta y las características de los ficheros que se tratarán a continuación, permiten definir claramente el área de operación de ficheros para cada usuario y/o proyecto.

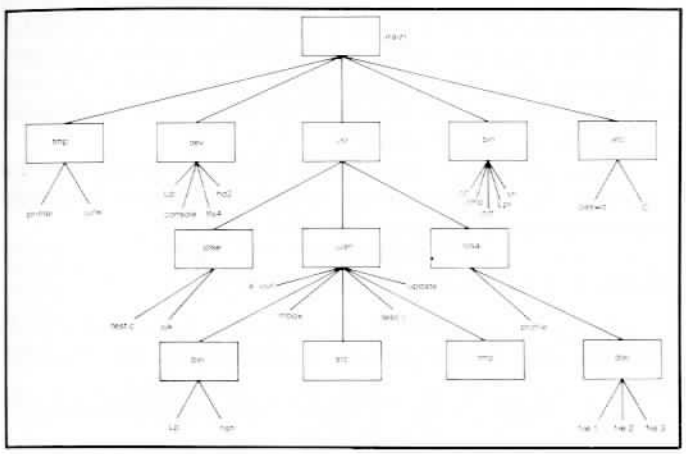

Fig. 2. Parte de un sistema de ficheros UNIX simplificado.

La referencia a un fichero se hace por una serie de nombres de ficheros separados por el símbolo «/», siendo el último de los nombres el del fichero al que se hace referencia. Si la cadena empieza por «/» quiere decir que la búsqueda empieza por el directorio raíz, si no la búsqueda empezaría por el directorio actual. En el caso de que se incluya el símbolo «../» la búsqueda debe empezar en el directorio del cual depende el directorio actual.

Por otra parte, no es preciso que todo el sistema de ficheros resida en el mismo periférico que el directorio raíz, para ello existe un comando de sistema («mount») que permite incorporar nuevas estructuras jerárquicas al árbol principal.

Además, también existe la posibilidad de que un mismo fichero físico esté referenciado bajo diferentes nombres, denominándose «linking», pudiendo de esta forma no tener que duplicarlos cuando son empleados en más de un área de aplicación y no existiendo, por tanto, el consiguiente problema de actualización simultánea de todos ellos. Un fichero de estas características sólo será borrado cuando no esté referenciado por ningún directorio.

La protección de los ficheros está representada por 10 bits en la referencia del directorio, lo que significa que existen protecciones diferentes según la cadena de directorios por la que se acceda al fichero. El primer bit indica si el fichero es un directorio o no; este bit puede interpretarse como un bit que indica contenido más que como un bit de protección. Los nueve bits restantes están divididos en paquetes de tres, representando cada paquete a uno de los tres tipos de usuarios posibles (propietario, grupo o todos). El usuario propietario es el que ha creado el fichero, el grupo es el conjunto de usuarios que pertenecen a un mismo proyecto con un indentificador común y todos es el conjunto de todos los usuarios autorizados para el uso del sistema.

Los tres bits de cada paquete representan la posibilidad de leer, escribir o ejecutar el fichero para cada categoría de posibles usuarios. Como cada categoría de usuario está incluída en la siguiente, tal como se muestra en la figura 3, los tipos de usuarios más amplios serán los que tendrán mayor o igual número de restricciones que los menos amplios.

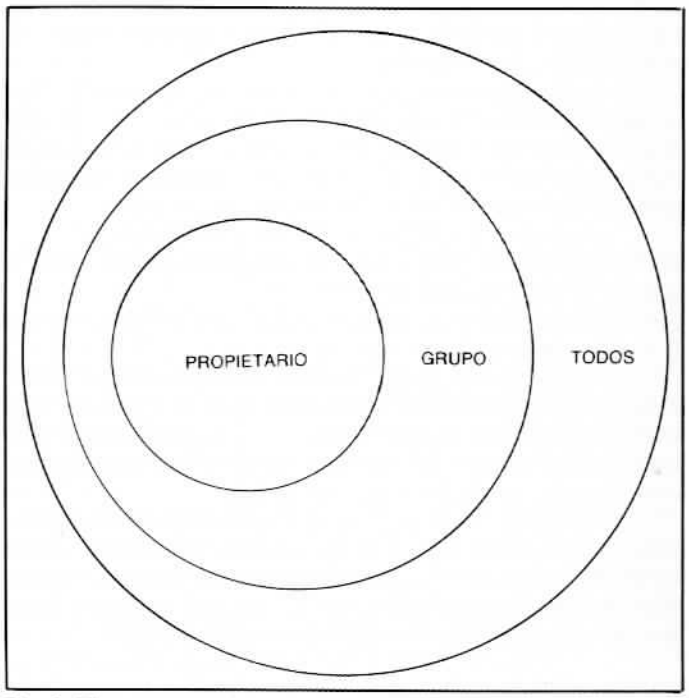

Fig. 3. Tipos de usuarios.

Para proteger la creación o borrado de ficheros de un directorio se efectua por medio de indicar la posibilidad de grabación en donde esté referenciado este directorio, lo que implica que la protección de borrado afecta a todos los ficheros miembros de un directorio.

Unicamente dos tipos de usuarios pueden modificar las protecciones de un fichero, el usuario popietario y el administrador del sistema (también llamado «superuser») utilizando el comando «chmod». Así mismo, es posible también definir las protecciones por defecto a asignar a todos los ficheros de nueva creación utilizando el comando «unmask» [6].

– El «shell»:

El «shell» es el mecanismo de comunicación entre el usuario y el sistema, no teniendo ninguna característica que lo diferencie de cualquier otro programa ejecutable, con lo que se le puede hacer un «swapping» de memoria a disco como cualquier otro proceso [3] [6].

El «shell» puede ser considerado tanto un interpretador de comandos interactivo como un lenguaje de programación [7]. Se puede considerar un lenguaje de comandos en el sentido clásico, ya que ejecuta líneas compuestas por un nombre de comando y unos parámetros.

El concepto de lenguaje de programación se le incorpora por las posibilidades de manipulación de variables, control del flujo de ejecución, acciones sobre la entrada/salida y opciones de ejecución.

Todos los aspectos concernientes a la gramática, metacaracteres y palabras reservadas del «shell» pueden encontrarse en la referencia bibliográica [7].

Cada programa ejecutado por el lenguaje de comandos del UNIX tiene asignados tres ficheros, uno de entrada, otro de salida y otro para mostrar los errores, utilizando todos ellos el terminal como periférico por defecto. Sin embargo, esta asignación puede ser modificada mediante los símbolos «<» y «>» de la siguiente forma:

COMANDO < ENTRADA > SALIDA

lo que significa que el proceso COMANDO toma los datos del fichero ENTRADA y deja los resultados en el fichero SALIDA.

El símbolo «> >» permite añadir al final de un fichero en vez de reemplazarlo. De esta forma simple se consigue redefinir la entrada y la salida sin tener que modificar el programa.

También son soportadas estructuras del tipo «if/then/esle», «case», «while», «for» y «until» para el control de la secuencia de ejecución.

Para construir nombres de ficheros o grupos de ellos se pueden emplear los símbolos «*», «?» y «[...]» que, respectivamente, representan cualquier combinación de caracteres, incluyendo el nulo, un carácter y cualquier carácter de un conjunto. De acuerdo con esto:

C son todos los ficheros del directorio tipo C.
A?.* es cualquier fichero de dos caracteres con el primero siendo A y de cualquier tipo.
BA[1-5] son todos los ficheros cuyo nombre empiece por BA y le siga un dígito que puede ser 1, 2, 3, 4 y 5.

Así mismo, se puede señalar que pueden definirse ficheros que contengan comandos, por lo que una línea a ejecutar puede empezar por un programa ejecutable que será un comando o por un fichero de texto que será considerado un fichero de comandos.

En la tabla 2 se incluye el repertorio de comandos más significativo [4]. Son de resaltar comandos como uniq, wc, grep y otros que convenientemente combinados dan una potencia considerable al sistema, ya sea en ficheros de comandos o en «PIPES» y «FILTERS».

at	Ejecuta un comando en un momento determinado
gal	Imprime el calendario
cat	Concatena e Imprime
cd	Cambia el directorio de trabajo
chmod	Cambia las protecciones de los ficheros
comm	Busca lineas comunes a dos ficheros ordenados
cp	Copia ficheros
crypt	Codifica y Decodifica información
date	Imprime la hora y la fecha
diff	Busca diferencias entre ficheros
du	Resume la utilización del disco
echo	Reescribe los argumentos
file	Determina el tipo de fichero
find	Busca un fichero en un directorio con condiciones especificas
grep	Busca las lineas con una cadena de caracteres en un fichero
kill	Termina un proceso
ln	Genera un «link» de ficheros
lpr	Pone un fichero en la cola de impresión del «spooler»
ls	Listado de un directorio
mail	Recepción y envio de mensajes a/o desde otros usuarios
man	Impresión selectiva del «UNIX Programmer's Manual»
mesg	Permite o Imprime mensajes
mkdir	Crea un directorio
mv	Traslada o Cambia de nombre a ficheros y directorios
passwd	Cambia la palabra clave de entrada del usuario al UNIX
PR	Imprime un fichero con paginación
ps	Estado de un proceso
pwd	Imprime el directorio actual
rm	Borra ficheros o directorios
sort	Ordena o fusiona ficheros
spell	Busca errores de escritura en palabras según un diccionario
stty	Poner las opciones del terminal
tail	Copia la parte inicial o final de un fichero
tee	Construye una «pipe»
tr	Cambio de caracteres con sustitución o eliminación
tty	Da el nombre del terminal
uniq	Elimina las lineas repetidas de un fichero ordenado
wc	Cuenta las lineas, palabras o caracteres de un fichero
who	Lista los usuarios, el terminal y la hora de entrada de los usuarios que están conectados al sistema
write	Escribe a otro usuario en otro terminal

Tabla 2.– Los comandos básicos del «shell».

También existe un conjunto de variables cuyo nombre empieza por «S» y que representan parámetros del sistema, desde el número del último proceso ejecutado en «batch» al carácter que representa la interrogación del «shell» por pantalla.

– «PIPES» y «FILTERS».

Los «PIPES» y los «FILTERS» son consideradas una de las más significativas aportaciones del sistema UNIX.

Un «PIPE» es un fichero abierto conectando dos procesos. Un «FILTER» es un programa que procesa una única cadena de entrada produciendo una única cadena de salida.

Por combinación de «PIPES» y de «FILTERS» es posible generar un nuevo comando llamado «PIPELINE». Los procesos en una «PIPELINE» se ejecutan concurrentemente y la sincronización, «scheduling» y el almacenaje intermedio de los «PIPES» es gestionado automáticamente por el sistema.

Para especificar una «PIPELINE» se especifican los nombres de los comandos del «shell» seleccionados, separándolos por el carácter «|». Por ejemplo:

who | wc – 1

da el número de usuarios del sistema.

Un ejemplo clásico es un programa verificador de la escritura de palabras de un fichero según un diccionario:

prep fichero | sort | uniq | comm diccionario

realiza las siguientes funciones:

- prep: toma las palabras del fichero y pone una en cada línea.
- sort: ordena las palabras.
- uniq: toma sólo las palabras diferentes.
- comm: compara con las palabras en diccionario e imprime las diferencias.

Todo esto hace que el «shell» por combinación con los comandos que posee con las opciones antes referenciadas, se convierta en una herramienta de programación simple, basada en la imaginación y la experiencia, pudiendo en algunos casos sustituir a los lenguajes convencionales.

– La Jerarquía de Procesos.

En el sistema UNIX la creación de un nuevo proceso se realiza mediante llamada a la primitiva, «fork», lo que provoca que el proceso actual se divida en dos procesos concurrentes, denominándolos «proceso padre» y «proceso hijo». Si bien estos procesos no comparten memoria principal, si que comparten todos los ficheros abiertos. Utilizando un sistema de «even-wait» es posible esperar que ocurra la terminación del «proceso hijo». Así lo hace «shell» para ejecutar comandos.

Hay que señalar que también es posible una ejecución en «batch», no teniendo que esperar el «shell» la terminación del «proceso hijo» que contiene el comando o la «PIPELINE». El símbolo empleado para ello es el «&».

Así por ejemplo:

(CC fuente; a. lis)&

compila y ejecuta en «batch» un fichero llamado «fuente».

Asociados con el «batch» existen un fichero «batch» de entrada y un fichero «batch» de salida, fichero de «login».

Así mismo, un proceso creado puede a su vez llamar al «shell» creando un «subshell» para ejecutar un comando.

La incorporación de cada usuario interactivo al sistema viene asociada a la creación de un proceso «shell» para el terminal por parte de UNIX. En la figura 4 queda reflejada la Jerarquía de Procesos [3].

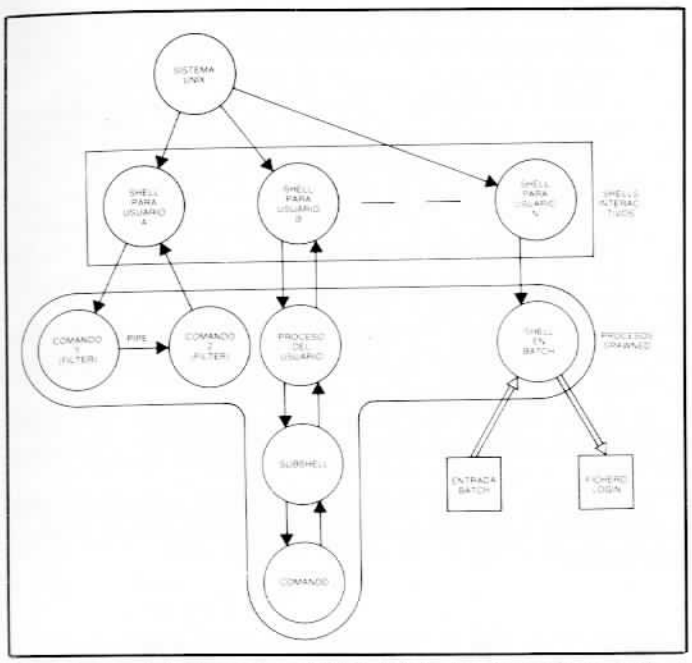

Fig. 4. Jerarquía de Procesos en UNIX. Las flechas descendentes indican notificación del final del proceso y final de la espera del proceso generador.

4. El UNIX como Entorno de Programación

El UNIX desde el punto de vista de Entorno de Prograación, al que de ahora en adelante se denominará E.P., está concebido como un repertorio de herramientas, en vez de como soporte de un método [1]. Esta filosofía implica la no obligatoriedad del uso de las mismas, es decir, no es necesario utilizar la metodología del sistema UNIX. La realidad es que con un simple aprendizaje se consiguen resultados bastante apreciables en la producción de programa, de forma que rápidamente la motivación a usar todas las herramientas de «software» va en aumento.

Las razones técnicas por las que UNIX es un E.P. están basadas, en parte, en los elementos descritos en la Introducción al Sistema y en especial en la estructura y tipo de ficheros, la potencia de opciones y estructura del «shell» y las «PIPELINES». Además, quedan un conjunto de herramientas que están más allá de la base del Sistema Operativo.

Estas herramientas son las organizadas paa la Preparación de Documentos [9], Herramientas de Desarrollo de Lenguajes [10] y el «Programmer's Workbench» [11].

– *La Preparación de Documentos.*

Uno de los elementos principales del E.P. es la preparación de textos, ya sean fuentes de programas, datos para los mismos, documentación de proyectos, comunicados, artículos o simplemente correspondencia y notas.

UNIX tiene las herramientas para obtener una sofisticada preparación de documentos. Este conjunto de herramientas incluye un Editor de textos, formateadores de textos programables, paquetes de definición de macroordenes para varios estilos de formateado y otros programas de soporte como los detectores de escritura correcta.

El Editor de textos se llama «ed» y está orientado a línea, debido a que los periféricos de la época en que apareció el UNIX eran de este tipo. Hay que señalar que existen otros orientados a «full-screen». Este editor no está orientado a ningún tipo de texto y no contiene ningún tipo de comando diferenciable de los editores usuales.

Existen dos módulos complementarios al «ed» que son el «grep» que busca líneas con una construcción determinada de serie de caracteres y el «sed» que es una variante de «ed» que ejecuta un conjunto de operaciones de edición sobre cada línea de una secuencia de entrada de longitud variable.

El formateado de los textos se realiza mediante «troff» y «nroff» cuya diferencia básica está en el tipo de periférico para el que generan su salida. El primero lo hace para máquinas de fotocomposición editoriales y el segundo para máquinas de representación de caracteres del tipo ASCII. Dentro de estas dos categorías se puede preparar salidas para periféricos concretos (impresoras de calidad, terminales gráficos,...). La entrada para «troff» y «nroff» está compuesta por líneas de texto entremezcladas con líneas de control.

Los paquetes de macros permiten incorporar fácilmente notas al pie de página, encabezados o estilos determinados según los macros utilizadas, entre otras posibilidades.

Además existe un juego de preprocesadores que pueden ser incorporados como «PIPELINES». El procesador «eqn» permite preparar expresiones matemáticas para incorporarlas al texto y «tbn» realiza una función similar con tablas.

En la práctica esta colección de utilidades ha demostrado ser de fácil aprendizaje y utilización, incluso para secretarias, tipógrafos y no especialistas. Por otra parte, existe el programa «learn» de enseñanza asistida por ordenador, donde hay muchos módulos orientados a mostrar la utilización de los elementos mencionados.

La experiencia ha demostrado que con este sistema se pueden preparar complicados documentos en la mitad de tiempo que con sistemas convencionales.

Por otro lado, la preparación de documentos se beneficia del hecho de que sea UNIX un sistema operativo de propósito general.

UNIX contiene también un conjunto de herramientas para el proceso estadístico de textos a nivel de carácter, cobinaciones de caracteres y palabras [13].

– *Herramientas de Desarrollo de Lenguajes.*

El desarrollo de nuevos programas en el sistema UNIX está facilitado por herramientas para el diseño y construcción de lenguajes. Estas herramientas son generalmente generadores de programas, compilables en C, las cuales suelen incluir algoritmos eficientes y no están concebidas para tareas determinadas.

Las dos herramientas principales de este tipo son «YACC» y «LEX». «YACC» es un generador de exploradores para gramáticas tipo «LALR(1)». «LEX» es un generador de reconocedores de expresiones regulares empleando autómatas finitos deterministas.

«LEX» y «YACC» pueden combinarse para generar una entrada totalmente analizada como muestra la figura 5.

Estas herramientas han sido empleadas en una extensa variedad de aplicaciones, desde compiladores, calculadores de sobremesa, lenguajes de formateado, hasta procesadores de formas. Cabe citar que el analizador de comandos del «shell» ha sido generado a través de ellas. Como resumen se puede decir que son herramientas lo suficientemente potentes para que su aplicación sea inmediata.

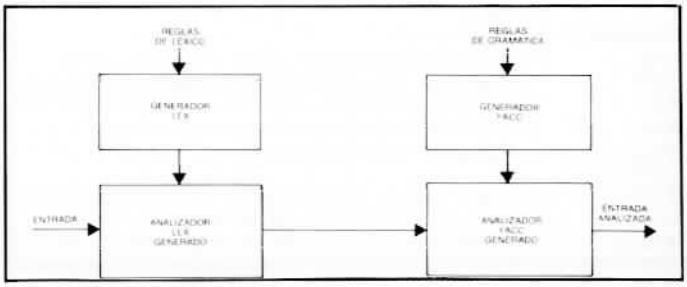

Fig. 5. Generación conjunta de «LEX» y «YACC».

– *El «Programmer's Workbench» (PWB/UNIX)*

El PWB es una versión expandida del sistema UNIX estandard especializada en el soporte de desarrollo de programas. El sistema gestiona proyectos de desarrollo desde uno a varios centenares de usuarios.

Ha demostrado ser lo suficientemente práctico para los desarrolladores de «software» como para que muchos de ellos abandonen las fichas perforadas y las copia de seguridad en cinta en los grandes «mainframes».

Se ha respetado en él la estructura modular y combinable del UNIX, además de su simplicidad, generalidad y fácil utilización.

El PWB permite separar de forma clara el concepto de sistema de desarrollo y sistema de ejecución, puesto que el PWB cubre las funciones del primero. Los sistemas donde se ejecutarán las funciones pueden ser variables, estando el PWB preparado para soportarlos.

Se puede decir que el sistema de desarrollo PWB ejerce las funciones de «front-end» respecto a los sistemas de ejecución, pudiendo ser instalado en sistemas de tamaño medio aunque los sistemas de ejecución puedan ser de tamaño superior.

El PWB puede también estar integrado por una red de sistemas de desarrollo interconectados.

Las funciones del PWB están soportadas por cuatro herramientas:

- Un soporte de preparación de documentos que básicamente es idéntido al del sistema UNIX estandard con algunas macros de forateado específicas.
- El «RJE» (Remote Job Entry) que permite enviar o recibir tareas de diversos computadores de ejecución y recibir las salidas de las tareas ejecutadas. El RJE se comporta para el «mainframe» como una lectora de fichas perforada y una impresora. Cuando se recibe el resultado de una ejecución, éste es grabado en el sistema de ficheros de UNIX, notificándose al usuario la llegada del mismo.
- El «SCCS» (Source Code Control System) mantiene todas las fuentes y los textos del sistema de desarrollo de «software». Así mismo, permite el almacenaje, actualización y recuperación, por número de versión o fecha, de todas las versiones de los módulos fuente o documentos. Todo cambio realizado en el «software» es registrado indicando quien, cuándo y por qué se ha realizado.
- Un sistema de prueba que permite realizar test en los sistemas de ejecución bajo el control de terminales en los sistemas de desarrollo.

Control de usuarios:	
ac	Informe del tiempo acumulado de conexión por usuario
sa	Informe de la utilización del «shell»
Comunicaciones:	
cu	Llamada a otro sistema en tiempo compartido
uucp	Copia de sistema UNIX a sistema UNIX
Herramientas básicas de desarrollo de programas:	
ar	Mantenimiento de archivos y librerias
as	Ensamblador
adb	Depurador interactivo
od	Volcado de ficheros
nm	Imprime la tabla de simbolos de un programa objeto
time	Estadistica de ejecución de un programa
Enseñanza asistida por ordenador (CAI):	
learn	Interpretador de comandos de CAI (aplicado a muchos elementos del UNIX)
Lenguajes:	
cc	Compilador de C
lint	Verificador de programa en C
cb	Indentación y correcta presentación del lenguaje C
f77	Compilador FORTRAN
raifor	FORTRAN estructurado
struct	Estructura programas en FORTRAN
Compilador de compiladores:	
yacc	Sistema de escritura de compilados del tipo LR (1)
lex	Generador de analizadores de léxico
Preparación de documentos:	
ed	Editor interactivo contextual
spell	Busca errores de escritura
typo	Sistema estadistico de búsqueda de errrores de escritura
Formateado de documentos:	
troff	Formateador para sistemas de fotocomposición
nroff	Formateador para periféricos de salida ASCII
ms	Generador de manuscritos estandard
eqn	Preprocesador de expresiones matemáticas
tbl	Preprocesador para tablas
Gráficos:	
graph	Prepara tipos de gráficos de un conjunto de puntos
spline	Sistema de interpolación de curvas
plot	Representación de gráficos en diferentes equipos

Tabla 3.– Utilidades de UNIX (lista parcial).

Hay que señalar que para el correcto funcionamiento del PWB se han mejorado las condiciones de fiabilidad del UNIX.

En la tabla 3 están referenciadas otras utilidades que completan la estructura del UNIX como Entorno de Programación.

5. Por qué y Para qué del Sistema Operativo UNIX

Para responder a estas preguntas hay que recurrir a la filosofía de diseño empleada [3]. La primera versión de UNIX fue construida en pocos años de trabajo de dos personas y los constructores eran básicamente usuarios de sistemas antes que diseñadores de sistemas, lo que hizo que el proyecto tuviera una gran coherencia.

El núcleo debía ser lo más simple posible pero con las suficientes funciones como para que combinándolas se pudiera desarrollar cualquier tipo de proyecto.

Debía de tener la mayor generalización posible y no repetir funciones específicas para cada tipo de elemento, como ocurre con la manipulación de ficheros, periféricos y memoria que están igualmente considerados.

Por último, se debía de poder realizar grandes procesos por combinación de los programas existentes antes que por construcción de nuevos programas, es decir, mejorar la eficiencia de desarrollo de programas.

Todo ello solucionaba los problemas de los Laboratorios Bell que se encontraban con una gran diversidad de equipos con varios Sistemas Operativos y de distinto tamaño, aplicados al desarrollo de programas y su documentación y a investigaciones de nuevas instrumentaciones.

– *La expansión del UNIX.*

Hay varios factores que han influido e influyen en la rápida y creciente proliferación de equipos con el Sistema Operativo UNIX. Mientras que algunos de ellos son factores tecnológicos, otros están basados en el entorno sociológico del momento. A continuación se exponen estos factores en base a un orden estrictamente cronológico.

La primera influencia que sufre UNIX es su generación en el lenguaje C, lo que hace que al sistema se le dote de una gran transportabilidad, así como una gran claridad en la comprensión de la estructura interna del Sistema Operativo. Dado que los fuentes son entregados con la licencia de explotación del UNIX se permite una fácil adaptación del sistema al usuario.

Laboratorios Bell distribuye el UNIX a varias universidades, sin coste o con un coste simbólico, lo que permite que sea experimentado por usuarios «duros» del Sistema Operativo. Por otra parte, según comentan Jensen y Wirth en el «Report del Pascal», el lenguaje más empleado es siempre el lenguaje más enseñado y viceversa, por tanto, si UNIX se emplea en las universidades su utilización se extenderá a las empresas. Además cabe señalar que

las universidades generan nuevos desarrollos que se incorporan al UNIX equipándolo con un gran número de herramientas.

Otra contribución importante la tuvo el hecho de que los ordenadores sobre los que se desarrollaron las primeras versiones de UNIX estaban muy extendidos en entornos de desarrollo de programas e investigación (máquinas DEC).

El nacimiento de los microcomputadores multiusuario, en los que cada vez es más fácil obtener en poco tiempo un «hardware» adecuado, pero no así un «software», abre un vacío cubierto por UNIX. Por otro lado, Laboratorios Bell, al no ser una empresa de ordenadores, no ejerce competencia ofertando UNIX en el mercado de «hardware».

La incorporación de Microsoft, conocedora del mercado de los micros, introduce el producto XENIX con una estructura más orientada a un tipo más amplio de usuario.

Todo ello contribuye a que prácticamente UNIX se convierta en un estandard para un conjunto de máquinas y aplicaciones.

– *El usuario UNIX.*

El sistema tiene sus limitaciones que configuran al tipo de usuarios. Los dos factores a considerar son la máxima potencia de opciones frente al mayor número de usuarios.

El sistema UNIX no pretende ser el más rápido, ni el más seguro, fiable y de máxima recuperación, como es usual en los grandes sistemas comerciales. Por ello, UNIX tiene su máxima eficiencia en organismos con diversidad de máquinas o aplicaciones que requieran una unificación de base del «software» con un proceso de formación simple y generalizado para todos los tipos de usuario.

En general se puede considerar al usuario de UNIX integrado en un equipo de desarrollo con una gran transportabilidad de «software» entre máquinas proyectos y organismos o bien usuarios que necesitan adaptar convenientemente el Sistema Operativo a sus proyectos, o en nuevas familias de mini y microordenadores como una herramienta potente y probada.

– *Los límites del UNIX.*

El sistema UNIX precisa una gestión de memoria potente, lo que hace que sean máquinas de 16 bits las máquinas más pequeñas en las que se ha utilizado. Las máquinas más grandes son aquellos «mainframes» tipo IBM 43XX que se encuentran dedicados, en algunos casos, al desarrollo de programas y entornos de investigación.

A pesar de su simplicidad en las encuestas no resulta ser el sistema operativo con un índice mayor de facilidad de aprendizaje e incluso se le considera no dirigido a los no iniciados en Informática. La realidad es que sus múltiples opciones y combinaciones le proporcionan una imagen ficticia de complejidad. Por otro lado, su generación hacia el principio de la década de los 70 hace que su estructura básica esté orientada a líneas aunque las nuevas utilidades ya lo estén a pantalla.

Entre las diversas propuestas de mejoras para UNIX se pueden citar las siguientes [12]:

- La manipulación de ficheros.
- Los mecanismos de protección y sincronización.
- La gestión de entrada/salida.
- La comunicación entre procesos.
- Las utilidades disponibles.

En la actualidad, en cada nueva versión de UNIX se van incorporando mejoras en base a las propuestas anteriores, siendo el UNIX SYSTEM V el que actualmente se distribuye.

Por otra parte, existen varios grupos de usuarios de los sistemas UNIX que centralizan las demandas de los usuarios e informan a sus miembros (USENIX, /usr/grup y UNI-OPS) [4] [6].

Teniendo en cuenta que UNIX no se encuentra en una fase de estabilización, sino que día a día se le incorporan nuevos conceptos, el «Computing Laboratory» de la Universidad de Newcastle ha desarrollado la «Newcastle Connection» [14] que permite no sólo la conexión entre varios sistemas UNIX de una forma transparente al usuario, sino que además permite un reparto de las cargas de forma eficiente entre los procesadores.

6. Conclusiones

La eficiencia del sistema UNIX en entornos de desarrollos de programas, núcleos de investigación de sistemas, organizaciones con necesidades de estandarización en el Sistema Operativo para diferentes máquinas y en pequeños micro y miniordenadores para acceder rápidamente a un potente Sistema Operativo ha quedado suficientemente probada.

Así mismo, la gran difusión del sistema UNIX lo ha convertido en casi un sistema estandard en el campo de los micro y miniordenadores de 16 y 32 bits. El concepto de estandar, sin embargo, queda algo difuminado debido a la facilidad de modificación al ser facilitadas las fuentes en lenguaje C. Este hecho ha motivado la existencia de diversas versiones del núcleo y del «shell», aunque en ellas siempre se respeta la filosofía básica para mantener la integridad del sistema.

Los E.P. reflejan el tamaño, organización y sociología de los grupos que los han creado, estando UNIX orientado en su filosofía básica al usuario típico de Laboratorios Bell.

UNIX resulta un E.P. sumamente adaptable debido a la facilidad de combinación de sus elementos básicos.

El E.P. también permite el soporte de diferentes niveles de experiencias, aunque por lo menos uno de los usuarios del sistema debe poder adaptarlo a cada necesidad.

UNIX soporta con diferente especialización todo el ciclo de vida del E.P., es decir, análisis, diseño, codificación, test y desarrollo, únicamente con herramientas de proceso de textos en algunas de las fases, mientras que en otras utiliza módulos integrados como el PWB.

Las herramientas del UNIX son módulos independientes en contraposición con la eficiencia de un sistema más integrado.

La especialización como E.P. es relativa. UNIX soporta varios lenguajes, pero resulta más efectivo con el lenguaje C y el «shell». No está orientado a ninguna metodología y soporta la producción de código para diferentes entornos de ejecución, pero de forma especial para los que utilizan UNIX. Existen compatibilidades con otros sistemas a nivel de comandos, llamadas al sistema y organización de ficheros.

En resumen, UNIX es un Entorno de Programación con un aceptable éxito tal como muestra su proliferación motivada en las razones expuestas.

Bibliografía

[1] P. Botella, H. Rodriguez «Ayudas a una Programación Sistemática: Entornos de Programación». RT.81/01 Facultat d'Informàtica (U.P.C.) (1981).

[2] B.W. Kernighna, J.R. Mashey «The UNIX Programming Environment». Software-Practice & Experience (Enero-1979).

[3] H.M. Deitel «An Introduction to Operating Systems». Addison-Wesley Publishing Company (1983).
[4] R. Thomas, J. Yates «A User Guide to The UNIX System». Osborne/McGraw-Hill (1982).
[5] R.C. Holt »Concurrent Euclid, The UNIX System, and Tunis». Addison-Wesley Publishing Company (1983).
[6] D. Fiedler «The UNIX Tutorial. Part 1: An Introduction to Features and Facilities. Part 2: UNIX as Applications-Programs Base. Part 3: UNIX in The Microcomputer Marketplace». Byte (Agosto, sept., Octubre-1983).
[7] S.R. Bourne «The UNIX Shell». Byte (Octubre-1983).
[8] S.R. Bourne «The UNIX Shell». The Bell System Technical Journal (Julio-Agosto 1978).
[9] B.W. Kernighan, M.E. Lesk, J.F. Ossanna jr. «Document Preparation», The Bell System Technical Journal (Julio-Agosto 1978).
[10] S.C. Johnson, M.E. Lesk «Languaje Development Tools». The Bell System Technical Journal (Julio-Agosto 1978).
[11] T.A. Dolotta, R.C. Haight, J.R. Mashey «The Programmer's Workbench». The Bell System Technical Journal (Julio-Agosto 1978).
[12] H. Broze, A. bruffaerts, M.F. Declerfayt y otros «Évaluation Critique du Systéme UNIX». T.S.I. – Technique et Science Informatique» (Vol. 1, n.º 4, 1982).
[13] L.E. McMahon, L.L. Cherry, R. Morris «Statistical Text Processing». The Bell System Technical Journal (Julio-Agosto 1978).
[14] K. Smith «Software Links UNIX Systems into One Unit». Electronics (19-Mayo 1983).