Sistema Operativo GNU/HURD

José Mariscal Prieto
http://www.uco.es/ i72maprj

14 de noviembre de 2003

Capítulo 1

Introducción

1.1. Historia

La historia sobre Sistemas Operativos comienza entorno a los años 50 se desarrollan Sistemas Operativos Monolíticos. Se trata de un diseño despreocupado sin estructura. Esta falta de estructura hace que necesiten hacerse mejores sistemas un poco mas complejos, necesitando una nueva estructura y organización. Sobre los años 70 se comienzan a desarrollar los primeros sistemas operativos multiusuario como como *Multics y después UNIX*, en esta época la mayoría de programadores compartían el código fuente, lo que hace que el desarrollo de una aplicación mejore notablemente. Además surgen nuevas técnicas de programación como la modularidad. Por otra parte estos sistemas necesitaban grandes y costosas máquinas, con lo cual solo estaban accesibles a un numero reducido de empresas y alguna universidad.

A finales de esta década y durante la década de los 80, cambia el modelo de mercado y las empresas empiezan a producir sistemas cerrados sin distribuir el código, como por ejemplo Digital en sus PDP, para impedir el uso de sus propias tecnologías en otros sistemas. Las computadoras de esta época tenían cada una su propio S.O. lo que impedía el desarrollo de una tecnología común, así como el desarrollo de drivers para una determinada arquitectura [1].

1.1.1. Proyecto Mach

Hace algunos años, entorno a 1.985 un grupo de personas de la Universidad de Carnegie Melon desarrollan un micronúcleo al que denominan Mach. La idea es básicamente crear un núcleo de sistema operativo en el que partes que están integradas en el núcleo, pasen a estar en el entorno de usuario, de esta forma se consigue un sistema con un mejor diseño que los núcleos monolíticos. El proyecto acabo en 1994 al moverse las vías de desarrollo de sistemas operativos en otras direcciones, de lo cual ha tenido una gran culpa que la mayoría de los núcleos que se han desarrollado desde entonces sean monolíticos.

[1] Es curioso como actualmente esto ocurre con algunos fabricantes de Hardware y Microsoft.

1.1.2. Proyecto GNU

Entre todo esto Richard Stallman un programador del MIT en el laboratorio de Inteligencia Artificial fue uno de los que trabajaron desarrollando aplicaciones y sistemas en los años 70. Al llegar la época de los 80 tubo un problema al querer imprimir en una impresora con drivers propietarios para cierto sistema, por lo que solicitó que le facilitaran información para hacer drivers para otra plataforma software para hacerla funcionar, no pudo hacer nada. Enojado el 27 septiembre de 1983 decidió anunciar la creación de un nuevo sistema llamado GNU (libre), una alternativa a los sistemas UNIX de pago, bastante caros por esta época. Aún así el concepto de Software Libre no quedó definido hasta enero del año siguiente:

- La libertad de usar el programa, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2).
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto.

Entiéndase que es un asunto referido a la libertad en cuanto a los usuarios y desarrolladores pueden hacer del código y no con su precio. Se confunde por el inglés (free) que significa tanto libre como gratis. Al tener este concepto mas claro Stallman funda la Fundación para el Desarrollo de Software Libre [2]

Mientras se comenzó a desarrollar software para este sistema GNU la parte del núcleo, que actualmente se conoce como GNU/HURD, se estaba retrasando por lo que decidió utilizarse un núcleo ya existente como *Linux* en el sistema GNU.

1.2. Breve historia de Hurd

La historia de *HURD* es un poco enrevesada, al principio en el año 1986 el núcleo que se estaba desarrollando para GNU era el *TRIX*, que fue desarrollado en el MIT de donde provenía Stallman. En diciembre de ese mismo año la *FSF* contacta con el profesor *Rashid* de la universidad de *Carnegie-Mellon* que estaba trabajando en el desarrollo del núcleo Mach. El trabajo consistiría en usar procedimientos de Mach los cuales servirían para arreglar o mejorar *TRIX*. Hasta 1990 no se hace desarrollo importante entorno al núcleo, por lo que se llega a adoptar el núcleo Mach para desarrollar lo que se conoce como HURD. GNU decide utilizar Linux como núcleo para funcionar sus aplicaciones, debido al inviabilidad de utilizar HURD por se demasiado inestable. Desde entonces hasta ahora se ha

[2] Free Software Foundation, a la que nos referiremos como FSF

trabajado en el desarrollo de HURD basándose en la versión 3.0 de dicho núcleo, actualmente se esta planteando mejoras de diseño en la versión 4.0 pero todavía no esta bien definida.

Capítulo 2

Micronúcleo

2.1. Micronúcleo

El micronúcleo surge como una nueva forma de organización para un sistema operativo, es un termino algo tedioso de entender ya que puede no ser relativo a su tamaño, pero si a su diseño. Alcanzó gran popularidad gracias a Mach, aunque existen otros como ,Minix, RC4000, Amoeba, Chorus y los basados en Windows NT (NT, 2000, XP y 2003).

En los sistemas monolíticos la mayoría de drivers se incluyen dentro de este [1], surge la idea de separar estas partes y ponerlas en el contexto de usuario mediante el uso de *Servidores*. De manera que las funciones básicas del núcleo son las que deben permanecer y las demás se implementan en el contexto de usuario. Esta implementación de SO hace relativamente sencilla la portación a diferentes arquitecturas como Alpha, X86 y otras.

La principal diferencia es el sistema de ficheros que en Unix a permanecido dentro del núcleo, ahora esto cambia y se implementa como un servidor de ficheros que se ejecuta en el contexto de usuario como un proceso mas.

2.2. Comparativa Micronucleo - Núcleo Monolítico

En esta sección examinaremos cuales son las partes principales de un sistema Monolítico a un Micronúcleo y como supera el Micronúcleo algunas problemáticas relativas al diseño de un sistema Monolítico.

Esta nueva arquitectura tiende a reemplaza los Sistemas Operativos en capas verticales por la horizontal en la que cada capa horizontal o algunas se implementan como servicios fuera del núcleo.

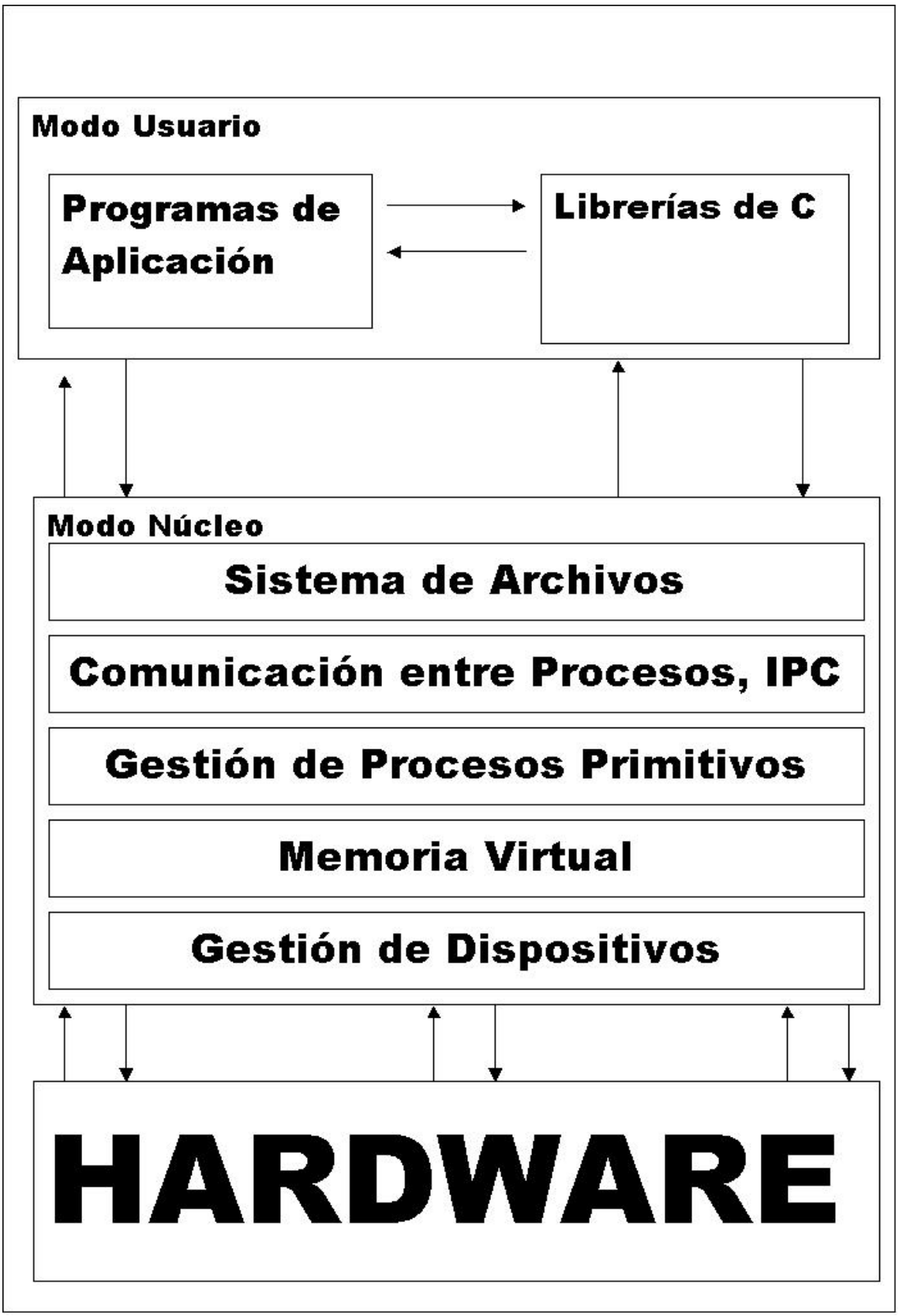
[1] Por ejemplo sistemas de ficheros como ext2 se incluyen dentro del núcleo

2.2.1. Definiciones

Antes de hacer la comparativa es necesario explicar dos términos importantes:

- Ejecución Modo Usuario
- Ejecución Modo Núcleo

La ejecución en modo usuario no necesita funcionalidades del núcleo, por ejemplo los programas de usuario. Pero cuando un programa necesita acceder al hardware lo hace por medio de un mensaje o llamada al núcleo y este es quien se encarga de ejecutarla, a esto se denomina ejecución en modo núcleo.



La imagen anterior muestra un núcleo monolítico por capas, la representación es muy básica, aunque vamos a comentar los aspectos que nos atañen. Normalmente en Unix se diferencian dos capas principales las que dependen del Hardware las que no. Entre las que dependen del hardware se encuentran:

- Memoria Virtual, se encarga de la gestión de la memoria a bajo nivel.
- Manejo de dispositivos, interrupciones.

La parte que no es dependiente del Hardware es común en casi todos los sistemas operativos monolíticos y sus principales componentes o funciones son:

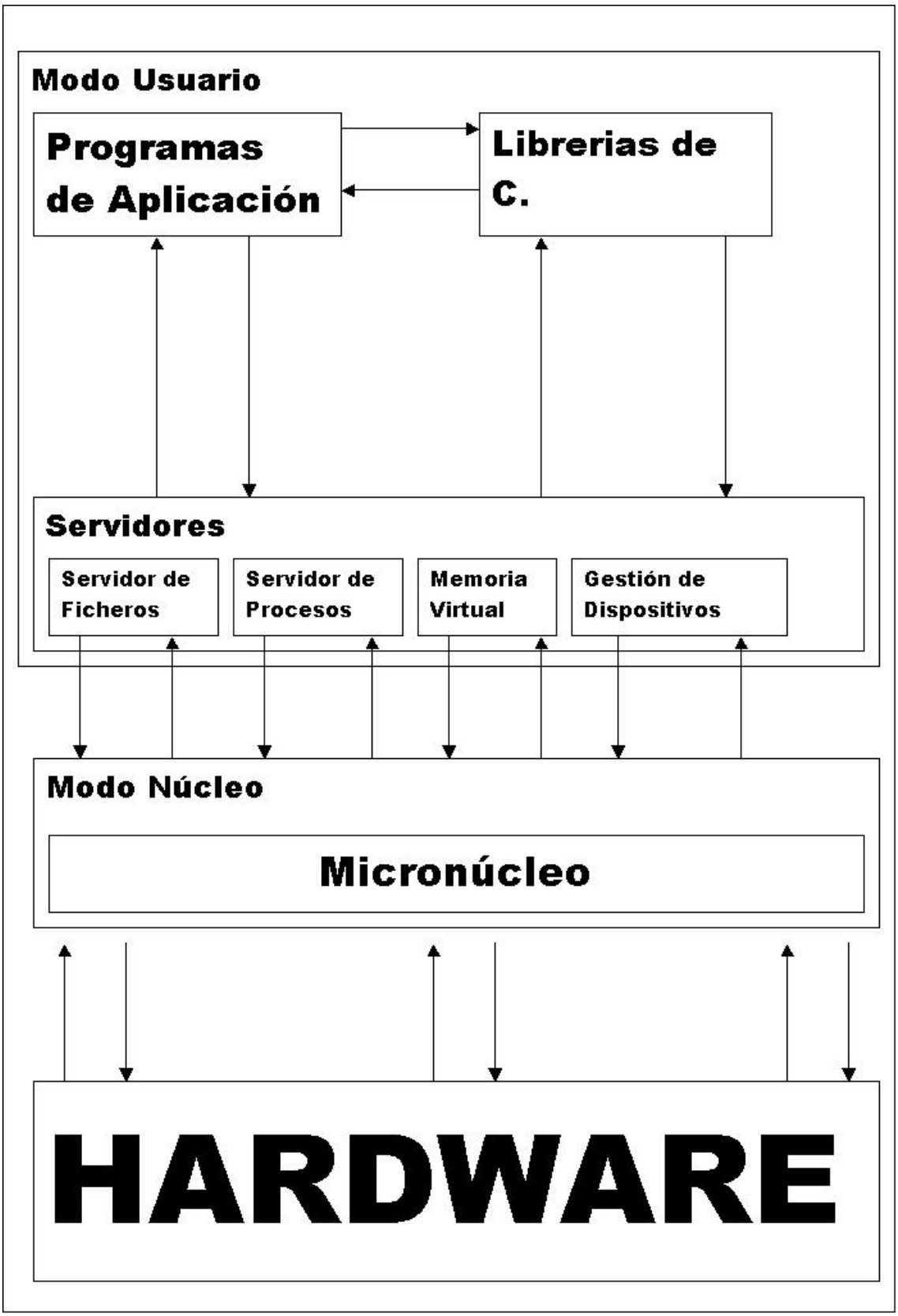
- Llamadas del sistema.
- Planificación de procesos.
- Manejo de señales.
- Sistema de archivos.

Esta parte aunque existan modificaciones no supone un trauma para crear nuevas funcionalidades o adaptarlas, mientras que un cambio a nivel del Hardware es necesita cambiar estructuras y empezar a reescribir código o empezar desde 0. Los sistemas Micronúcleo por contra intentan que esto sea lo menos traumático posible y hacer que se modifique lo menos posible el núcleo.

Existen una serie de problemas que son inherentes al diseño monolítico:

1. Si se modifica el Hardware por lo general es necesario recompilar el núcleo para poder disponer de las funcionalidades, un ejemplo de esto podemos verlo en Linux.
2. Si se necesita alguna funcionalidad como un nuevo sistema de ficheros es necesario recompilar el núcleo para que lo soporte o en caso de que se puedan utilizar módulos, cargar el módulo[2].

[2]Esto tiene cierta similitud con los servicios en HURD y la posibilidad de usar o no usar por ejemplo un servidor de ficheros



Como se puede apreciar de la imagen anterior los componentes que estaban dentro del núcleo en los sistemas monolíticos están ejecutándose como procesos en modo usuario.

La comunicación entre estos servidores es asíncrona y se realiza por medio de paso de mensajes. El micronúcleo se encarga de gestionar la comunicación entre los servidores y los programas que llamaremos cliente que serian los procesos de usuario tal y como los

conocemos en UNIX.

2.3. Implementación

El núcleo Mach se encarga también de gestionar los manejadores de dispositivos y la memoria virtual. Para poder hacer esto tenemos dos alternativas:

1. Implementar el manejador del dispositivo físico fuera del núcleo.
2. Añadir funcionalidades al núcleo para que provea de mecanismos para que los servidores puedan acceder.

La primera solución es muy costosa por lo que por ejemplo en el Mach 3 lo que se hace es que el acceso a los dispositivos se hacen desde el núcleo, los manejadores de dispositivo a nivel usuario van lentos. Por contra la solución 1 se esta planteando para implementarla en el L4 siguiente sucesor de la versión 3 que estudiaremos mas adelante.

Lo que si se hace fuera del núcleo y esta es la importancia de HURD es que los sistemas de ficheros se desarrollan todos en el espacio de usuario, aunque para esto se han necesitado otros tipos de servidores adicionales como son los *Traductores*.

La idea de proceso como tal en HURD no existe como tal, aunque si de manera lógica, mas bien sería Tareas y hilos. Una tarea se trata de un espacio donde se realiza la ejecución y un hilo es una entidad ejecutable. La tarea es la unidad básica por la cual el núcleo asigna los recursos, se agrupan entorno a un grupo de hilos. Comparándolo con UNIX un proceso seria una tarea con un solo hilo de ejecución. Para parecerse a un sistema UNIX lo que hace es usar un servidor de procesos, para que de forma transparente gestione las tareas como si se tratase de procesos.

Por ahora HURD sólo esta disponible en la versión 3 de Mach, ahora mismo se está planteando como va a ser el nuevo núcleo L4 que comentaremos mas adelante.

2.4. Ventajas de los Micronúcleos

Algunas ventajas que podemos destacar de los micronúcleos son las siguientes:

- Uniformidad de Interfaces: Disponen de una interfaz única para las solicitudes de los procesos, el paso de mensajes.
- Extensibilidad: Debido a que las capas ahora son verticales y son un servidor fuera del núcleo, añadir una nueva capa es mas sencillo que hacerlo en un sistema Horizontal

[3]

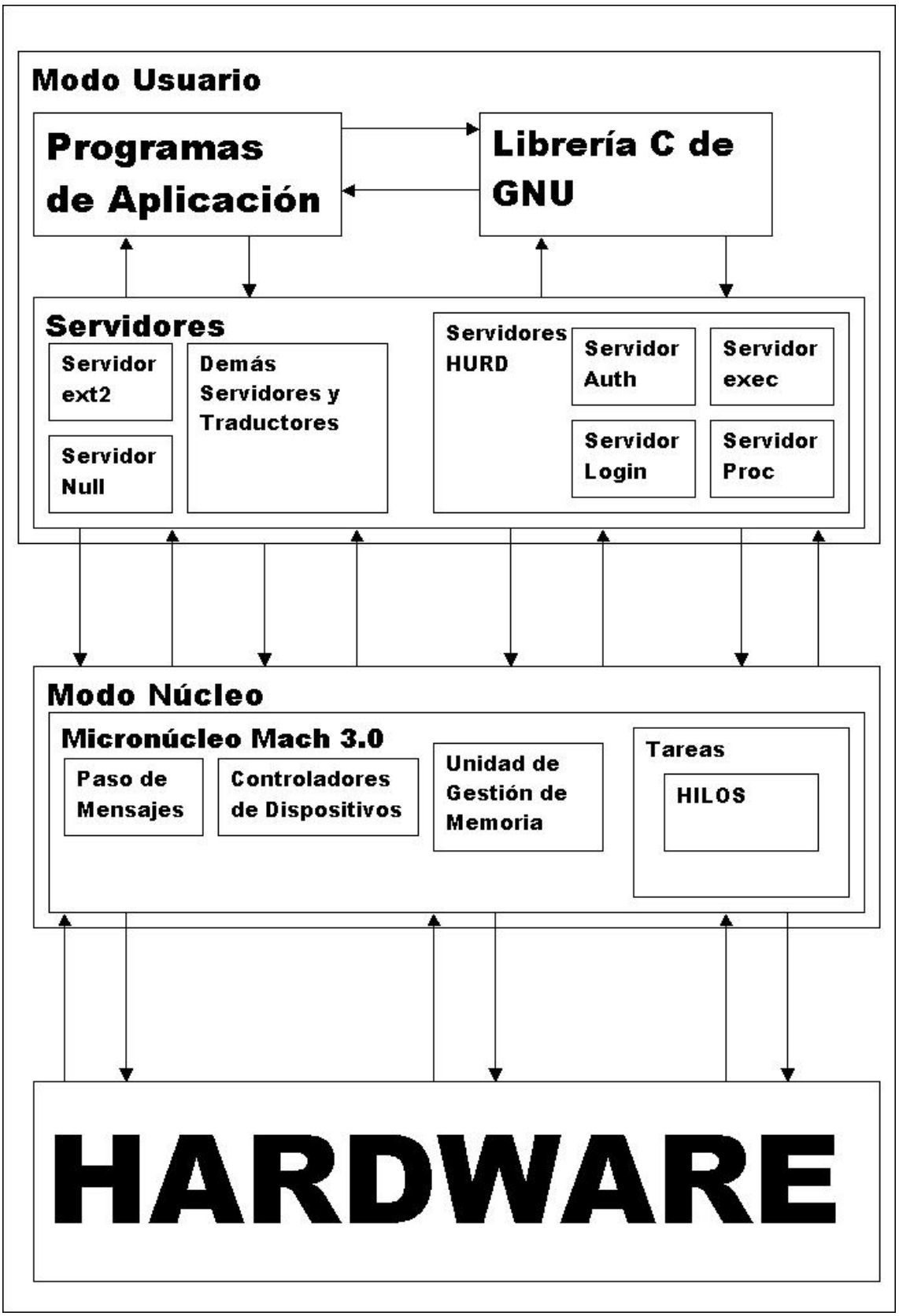
- Flexibilidad: Un efecto de lo anterior pero la manera de hacerlo ha de ser lo mas simple posible reduciendo características, haciendo una implementación mas pequeña y por tanto eficiente.
- Portabilidad: Reduciendo el núcleo y implementando casi todo en servidores, para implementarlo en arquitecturas diferentes solo habría que modificar el núcleo haciendo mas simple su portabilidad.
- Fiabilidad: Es mas fácil corregir fallos en un sistema pequeño y se pueden realizar pruebas más rigurosas que en un sistema mucho mayor.
- Soporte de Sistemas Distribuidos: Tales como NFS, FTP y otros sistemas de red, hacen por ejemplo que un servidor ftp sea accesible como un simple directorio.

2.5. Micronúcleo en HURD

Hurd es un modelo de Sistema Operativo basado en el micro núcleo GNUMach. HURD [4] significa Horda de Demonios que Reemplazan a Unix, estos Demonios son servidores de HURD, y su funcionalidad es reemplazar a los núcleos Unix.

[3] Nos referimos a los sistemas monolíticos por capas, que una nueva funcionalidad en capas cercanas a Hardware necesitarían cambiar todas las capas posteriores

[4] Hird of Unix-Replacing Daemons



La idea de este nuevo modelo es separar partes que estaban dentro del núcleo, sacarlas y ponerlas en el espacio del usuario, por ejemplo el sistema de ficheros. Por otra parte debido a la complejidad que suponía un servidor de gestión de memoria y de acceso a hardware, (una especie de servidor drivers que controlara la entrada/salida y interrupciones) era bastante costoso implementarlo así que se metió en el núcleo.

Cuando hablamos de Tarea en HURD nos referimos a la idea de proceso, pero con varios hilos de ejecución. Un hilo lo podemos definir como una entidad ejecutable dentro del proceso que normalmente depende de otros hilos en ejecución dentro de la tarea.

2.6. Demonios básicos de HURD

Los demonios son una serie de servidores que lo que hacen es sustituir gran parte del núcleo de UNIX y pasarlo al espacio del usuario, de manera que un demonio actúa como si fuese un proceso mas del sistema, por tanto se ejecuta en el modo usuario.

Los tipos principales de demonios en HURD son los siguientes:

- Auth Un servidor de Autenticación.
- Exec Un servidor de Ejecución.
- Proc Un servidor para el manejo de procesos (tareas) [5]
- Login Un servidor que se encargue del manejo de usuarios

Estos son los principales servidores, aunque existen muchos más que comentaremos después. Los servidores se comunican mediante pasos de mensajes a través de un puerto, esto hace posible la comunicación entre procesos.

[5] Cuando hablamos de Tarea en HURD nos referimos a la idea de proceso, pero con varios hilos de ejecución.

2.7. Hilos

Hay que reseñar que en HURD el concepto de proceso existe como el de Tarea formada por una serie de entidades ejecutables a las que se llaman hilos. Estas Tareas son gestionadas por el núcleo.

2.7.1. Definición

Una definición podría ser como conjunto de entidades ejecutables que se ejecutan al mismo tiempo dentro de un proceso, de manera que un proceso esta formado por uno o varios hilos, al menos es así como se implementan en la mayoría de Sistemas Operativos de hoy día, como Linux, Solaris, Windows 2k y por supuesto HURD. La ventaja de utilizar hilos es por ejemplo la posibilidad de que dentro de un proceso se pueda bloquear una seria de hilos, pero que sin embargo el resto continúen su ejecución. Además existen otras ventajas destacables del uso de hilos son las siguientes.

Los hilos están formados por:

- Estado de ejecución del hilo: Indica si el hilo esta en ejecución, listo o bloqueado.
- Contexto del procesador: Es importante salvar el estado del hilo cuando se bloquea, permitiendo luego restaurar los registros que estaban en el procesador para seguir su ejecución.
- Una pila de ejecución: Donde guardar los datos para saber por donde esta ejecutando.
- Almacenamiento estático para las variables locales, son las que están dentro del hilo.
- Acceso a memoria de los recursos del proceso: Se debe compartir y permitir el acceso a variables que comparten varios hilos del mismo proceso, surgen aquí los problemas típicos de interbloqueo y exclusión mutua que se aplican en procesos.

Frente a los procesos mono-hilo, los procesos multi-hilo presentan algunas ventajas:

- El tiempo en crear un hilo es menor que el empleado para un proceso.
- El tiempo para finalizar un hilo es menor que el tiempo para un proceso.
- Es más fácil cambiar información entre dos hilos que entre procesos.
- El empleo de hilos aumenta la eficiencia de comunicación entre programas, si los hilos están dentro del mismo proceso en ejecución la comunicación es mucho mas rápida.

2.7.2. Hilos en HURD

HURD es un sistema que utiliza sobre todo hilos. La comunicación entre hilos se realiza por pasos de mensajes. Podemos ver la comunicación como un puerto que va recibiendo una serie de mensajes, es decir una cola de mensajes, la cual debe gestionar el proceso. Para realizar esta tarea se utiliza el servidor de procesos, que lo que hace es suplir algunas características que en UNIX se implementaban en el sistema de ficheros, tales como el UID, GID que tenía un proceso. Sin embargo en HURD esto es diferente, estos permisos se asignan mediante puertos.

Mensajes

Un mensaje es una abstracción de Mach para permitir la comunicación entre procesos. Un mensaje es una estructura de datos formada por:

- Cabecera: contiene el tipo de contenido que tiene el mensaje, es decir si tiene una cadena de caracteres, enteros, etc...
- Datos: que es el contenido del mensaje.

Aunque para que un mensaje llegue a su destino es necesario ser enviado mediante un puerto.

Puertos

Un puerto se puede definir como un sistema que permite comunicar Tareas, recibiendo una serie de mensajes que son encolados para su tratamiento. Así pues un puerto es una estructura de datos dentro del núcleo. De esta manera un puerto tiene una serie de permisos definidos, lectura, escritura, etc... De manera que un hilo envía un mensaje a un puerto de otro hilo, además se implementan mecanismos por ejemplo para que el hilo emisor no se bloquee en espera de una respuesta.

Los puertos solo pueden ser manejados por el núcleo, Las Tareas deberán referirse a un nombre de puerto que son dependientes de cada proceso o hilo. Por tanto un puerto puede

tener nombres diferentes según la Tarea que tenga acceso a ese puerto. Así que un puerto esta formado por:

- Un nombre de puerto: relativo a las tareas que accedan.
- Derechos de puerto (port right): que define los derechos de acceso al puerto que pueden ser:
  - Derechos de envío (send right): definen los permisos para el envío de mensajes.
  - Derechos de recepción (receive right): Que definen los derechos para recibir mensajes.

De esta manera un puerto puede tener muchos permisos de envío pero solo un derecho para recepción.

HURD necesita además un espacio global de puertos para que los hilos no se comuniquen solo dentro de una misma tarea. LA comunicación global entre tareas requiere que el sistema tenga un registro global de nombres de puertos para que las tareas puedan acceder a estos.

2.7.3. Similitudes con UNIX

Podemos hablar de algunas similitudes con UNIX que pasan a estar implementadas en HURD de diferente forma, en parte por separar los sistemas de ficheros ponerlos como un proceso mas en el espacio de usuario. Veamos algunas de estas similidutes:

1. En Unix los procesos acceden a los ficheros mediante un descriptor de ficheros [6] que recuerda a los puertos en HURD.
2. Un solo fichero (permisos de recepción) puede ser abierto por varios procesos (permisos de envío).
3. Los descriptores de fichero son nombres locales a cada proceso (nombre del puerto).

[6] Existen tres descriptores: stdin, stdout, stderr

Capítulo 3

Servidores

3.1. Servidores del sistema HURD

En Hurd existen varios tipos de servidores:

3.1.1. exec

Este servicio se encarga de la creación de las imágenes de los procesos a partir del código objeto, este código objeto puede ser un fichero compilado por ejemplo ELF, a.out o ejecutable comprimidos con gz [1] (utilizados en el arranque) Digamos que se trata de una versión reducida del planificador de UNIX que se encarga de cuando hacemos una llamada fork().

3.1.2. init

Es el servidor que se encarga de que programas ejecutar cuando arranca el sistema, así como parte de su configuración. De manera análoga se parece al proceso init que crea todos los procesos en UNIX.

3.1.3. auth

Este servicio sirve para cuando dos servidores no confían uno en el otro. Entonces pasan la petición al server auth y este trata de ponerlos en contacto estableciendo un dominio de confianza. Esto hace posible que dos procesos tenga varias identidades al mismo tiempo, obteniéndolas y recuperándolas de forma dinámica.

[1] gz es el tipo de archivo comprimido en UNIX gzip

3.1.4. proc

Este servidor se encarga de asignar los PIDs y las estructuras de procesos a las tareas, crea lo que conocemos contexto de un proceso. Se encarga también de gestionar parte de las llamadas fork() y soporte de la biblioteca de C. Entiéndase proceso en HURD como una TAREA formada por una serie de hilos, es un concepto un poco diferente al de UNIX por la utilización de mensajes.

3.1.5. crash

Este servidor se activa cuando una tarea recibe una señal de error fatal, por ejemplo se ha accedido a una zona de memoria que no es accesible [2]. El servidor puede hacer varias cosas cuando esto ocurre, suspenderlo, matarlo o crear una imagen en memoria (aún no implementado).

3.1.6. ext2

Se encarga del manejo de sistemas de archivo con Extendido 2. Hace lo mismo que ext2fs.static, salvo que este esta enlazado de forma estática lo que lo hace imprescindible para poder montar una partición raíz con ext2fs, o arrancar el sistema en modo monousuario, por tanto es útil si se quiere arrancar un sistema instalado con HURD que este servicio se inicie.

3.1.7. ftp

Un servidor para los sistemas de archivo ftp. La utilidad de este servicio es poder por ejemplo montar un sitio ftp y trabajar como si se tratase de un directorio del sistema, pudiendo crear ficheros, directorios o borrarlos como si se tratase de uno mas del sistema.

3.1.8. isofs

Un servidor para los sistemas de fichero iso, por ejemplo iso9660 utilizado en los discos compactos.

[2] Un fallo de segmentación. Segmentation fault

3.1.9. fwd

Lo que hace es reenviar las peticiones a otro servidor. Se utiliza con fifo y symlink. La utilidad es poder utilizar un servidor de puente, y así no tener que crear mas servidores adicionales para tareas comunes.

3.1.10. nfs

Soporte para el sistema de archivos de red de Sun.

3.1.11. null

Este servidor lo que hace es implementar el /dev/zero salida de 0 infinitos y /dev/null (el espacio infinito para dejar datos).

3.1.12. pfinet

Es un servidor para TCP/IP, que implementa los protocolos IP versión 4.

3.1.13. symlink

Servidor de enlaces simbólicos para sistemas de ficheros que no los implementan.

3.1.14. term

Servidor de terminal, implementa una terminal al estilo POSIX. Por ejemplo una tty o una pts.

3.2. Traductores

La idea de traductor es por ejemplo que en UNIX hay archivos de dispositivo especiales que sirven para comunicarse con el núcleo, además de otros mecanismos como colas, enlaces fijos. Todos estos objetos son diferentes pero tienen algo en común, como propietario, derechos de acceso. Para añadir un objeto por ejemplo en sistemas como UNIX es necesario

modificar el código existente. Hurd provee de una interfaz general para añadir nuevas funcionalidades, para ello lo que se hace es introducir un programa que se inserte entre el contenido real y el usuario que lo solicita, a esto es lo que se denomina *traductor*.

Un traductor se ve como un proceso de usuario, de manera que estos los puede ejecutar cualquier usuario. La información sobre los traductores se almacena en un inodo.

Existen dos tipos de traductores:

- Activo: Se trata de un proceso traductor en ejecución.
- Pasivo: Los traductores pasivos se inician cuando es necesario.

De manera que un traductor pasivo solo se inicia cuando por ejemplo se monta una partición cuando se acceda (similar al automounter). O activar la red cuando se utilice.

3.2.1. fifo

El Traductor fifo[3] implementa una tubería, útil para implementar buffers

3.2.2. new-fifo

Crea una nueva cola fifo.

3.2.3. firmlink

Traductor de enlaces fijos, similar a los enlaces duros, *hardlink*, de los sistemas UNIX.

3.2.4. magic

Este traductor devuelve información sobre el proceso llamante, se implementan bajo la biblioteca de C.

3.2.5. ifsock

Se encarga de gestionar los nodos de sistemas de archivo, pero sólo en sistemas que no lo hacen por si mismo. Los que hace es actuar como enlace para sockets de UNIX.

[3]Fist In Firs Out, primero en entrar primero en salir.

3.2.6. storeio

Es un traductor de dispositivos ya sean de almacenamiento o de comunicación.

Capítulo 4

L4

4.1. Mach L4

El L4 es la siguiente versión para el núcleo Mach, hasta ahora GNU/HURD utiliza el Mach 3, pero debido a recientes cambios y las nuevas líneas de investigación entorno al L4 el proceso de crear un SO bajo el núcleo Mach 3 ha sido prácticamente congelado, y se esta investigando entorno a desarrollar un nuevo kernel, con similitudes al 3, pero con dos novedades principales.

- Ejecutar en el entorno de usuario, el acceso a interrupciones E/S utilizando mecanismo de comunicación con el núcleo.
- Ejecutar un sistema de gestión de memoria virtual en el entorno de usuario.

Estas ideas han hecho que algunas partes del L4 se empiecen a desarrollar prácticamente desde cero.

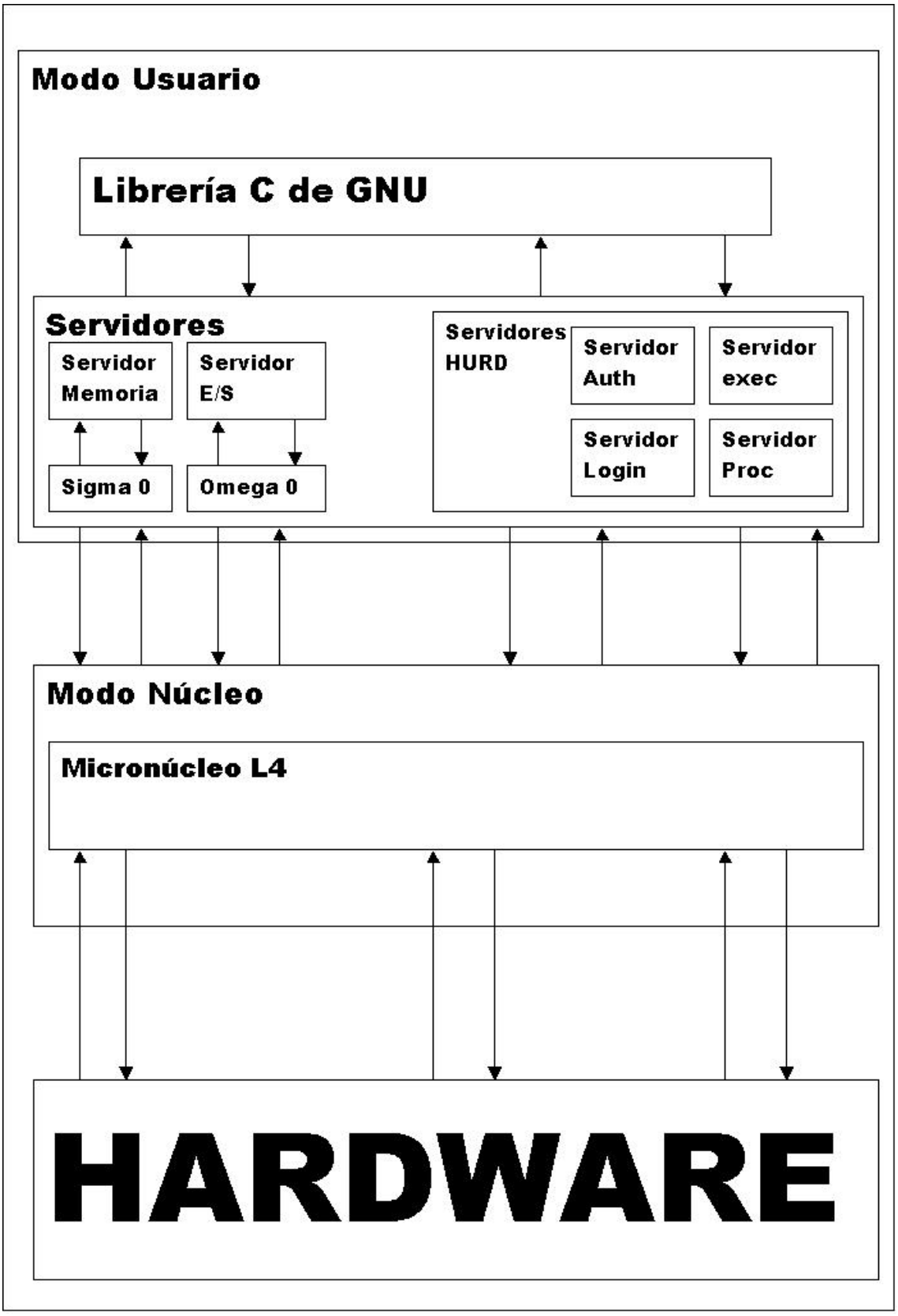
4.1.1. Nueva Organización.

Otra diferencia de L4 frente a su anterior Mach 3 es que la gestión y el acceso a los dispositivos se hace mediante un servidor y no se encarga de hacerlo el núcleo. El micronúcleo solo proporciona primitivas básicas para el acceso al Hardware.

Algunas novedades que implementa L4 son las siguientes:

- Servidor de Memoria. Sigma 0
- Servidor de Entrada/Salida. Omega 0
- Servidores de HURD: auth, login exec y proc, aunque estos estaban ya en Hurd.

El Servidor de Memoria implementa un sistema de memoria virtual, para ello se ayuda del Sigma 0 que hace las llamadas al núcleo. El Servidor de Entrada/Salida se ayuda del Omega 0 que es otro servidor que realiza las interrupciones.



Los Servidores HURD, aquí existe un problema para implementar el paso de mensajes,

IPC [1]. El principal obstáculo para el IPC es que la comunicación entre Tareas es Asíncrona.

4.1.2. Servidor Sigma0

Este servidor se encarga de manejo de primitivas de la memoria desde el punto de vista físico. Solo provee primitivas el que hace realmente todo el trabajo de gestión de memoria a un nivel mas alto es el Servidor de Memoria, que implementa la gestión de memoria virtual a nivel de sistema operativo encargándose de como deben los procesos acceder a la memoria.

4.1.3. Servidor Omega0

De manera análoga, Omega0 provee las primitivas para comunicarse con el hardware, solicitud de interrupciones y registrar dispositivos nuevos. La idea es implementar un entorno para desarrollar drivers en el entorno de usuario de manera que si se quiere añadir un nuevo dispositivo sea relativamente sencillo, lo único que habría que hacer seria portar las estructuras de datos de acceso para el Servidor de Entrada/Salida, este se comunicaría con el Servidor Omega0 para registrar un nuevo dispositivo o acceder a este, no siendo necesario acceder al dispositivo a bajo nivel.

[1] Iter Proccess Comunication, Comunicación Entre Procesos

Capítulo 5

Práctica

5.1. Instalación

Para instalar hemos utilizado la distribución inestable de Debian basada en núcleo HURD 3.0.24-GNU-K.

Por ahora la instalación de HURD no se hace ejecutando un núcleo HURD desde el principio, como es inestable, utiliza un núcleo Linux, y lo que hace es crear un sistema de ficheros ext2 y copiar ahí el sistema base. [1] Por tanto es altamente recomendable tener instalado Linux para poder instalar y acceder a HURD mas fácilmente.

5.2. Arranque

Para arrancar es necesario utilizar un programa cargador llamado GRUB, capaz de pasarle parámetros al núcleo como que módulos tienes que cargar, con módulos nos referimos a los servicios que HURD debe iniciar en el proceso de arranque.

Para arrancar HURD necesitamos pasarle ciertos parámetros.

```
# For booting the GNU Hurd
title  GNU/Hurd
root   (hd1,6)
kernel /boot/gnumach.gz root=hd1s7
module /boot/serverboot.gz
```

Comentaremos un poco las opciones de GRUB

[1] Una versión reducida de los programas del SO, capaz de crear dispositivos, y instalar nuevos programas

- `root (hd1,6)` Indica que la partición con el sistema de ficheros raíz está localizada en el primer Ide como esclavo, y está en la partición 7. Lo que se hace es decirle a GRUB donde tiene que buscar el núcleo.
- `kernel /boot/gnumach.gz root=hd1s7` le dice a GRUB el directorio donde se encuentra el kernel y además le pasa el parámetro para montar el sistema de ficheros raíz.
- `module /boot/serverboot.gz` Aquí le decimos al núcleo que debe cargar el ejecutable serverboot.gz que lo que hace es arrancar los servidores en el inicio.[2].

Existen otras maneras de arrancar pasando parámetros y diciéndole a GRUB que servicios adicionales debe cargar por supuesto las ordenes son demasiado complicadas y de varias líneas, que serverboot.gz se encarga de suplirlas perfectamente. Una vez hecho esto HURD arranca, es importante no tocar el teclado ya que existe un bug que hace que nuestro sistema se cuelgue y por tanto HURD reinicie.

5.3. El Sistema

Una vez arrancado el sistema, la interfaz es similar a Unix, inicia el bash y no hay nada configurado. Por ejemplo en Linux cualquier sistema trae el directorio /dev con los dispositivos creados, lamentablemente en HURD no, por lo que si queremos poder montar una partición del disco duro o una unidad de CD deberemos crear el dispositivo manualmente. Para configurarlo primero hay que ejecutar:

```
export TERM=mach
```

y es importante hacerlo ya que luego si ejecutamos native-install que es el siguiente paso, entra en un bucle (configurando la zona horaria) del que no puede salir y como es la única terminal no es posible continuar. Ejecutamos el script, nos pide que configuremos la zona horaria, una vez acabado hemos de reiniciar otra vez y volverlo a ejecutar.

Antes de hacerlo le echamos un vistazo a los programas en ejecución con ps:

```
USER        PID %CPU %MEM     SZ    RSS TT STAT    START     TIME COMMAND
0             0  0.0  0.1   130M   768K  ? R<mo   8:03PM  0:00.03 /hurd/proc
-             2  0.0  1.7   803M 12.8M  ? D<p    8:03PM  0:00.01 -x
0             3  0.0  0.1   1.1G 1.05M  - So     8:03PM  0:00.10 /hurd/ext2fs.
0             4  0.0  0.1   130M  936K  - So     8:03PM  0:00.05 /hurd/exec
0             5  0.0  0.1   130M  688K  - So     8:03PM  0:00.00 /hurd/auth
0             7  0.0  0.1   130M  840K  - So     8:03PM  0:00.03 /hurd/term /t
0            10  0.0  0.1   145M  656K  - S      8:04PM  0:00.01 ps -aux
```

[2]Pueden ejecutarse manualmente pasando una complicada línea al GRUB, tal y como muestra el manual de usuario

Volvemos a reiniciarlo y ejecutamos el script. Terminamos y volvemos a reiniciar. Ahora nos aparece una terminal para hacer login, pero es un login diferente al de los sistemas Unix clásicos. el prompt aparece así:

```
LOGIN>
```

Así que para entrar hacemos *login root* y ya entramos en el bash[3].

Probamos a hacer un echo hola > /dev/null y efectivamente se activa el servidor null.

Tambien probamos a ver que particiones hay montadas con mount

(Segmentation Fault)
Lo ejecutamos nuevamente y el sistema se queda bloqueado.

Reiniciamos, creamos nuevos dispositivos hd2 hd3 y podemos acceder al cdrom, montamos y utilizamos dselect para instalar los paquetes del cd ya que apt-cdrom add falla.

Hacemos un ps para mostrar que hay ejecutándose ahora...

```
USER       PID %CPU %MEM    SZ   RSS TT STAT   START     TIME COMMAND
root         0  0.0  0.1  130M  848K  ? R<mo 10:04PM  0:00.07 /hurd/proc
-            2  0.0  1.7  803M 13.1M  ? D<p  10:04PM  0:00.02 -x
root         3  0.0  0.3  1.1G 2.28M  - So   10:04PM  0:00.61 /hurd/ext2fs.
root         4  0.0  0.1  130M  956K  - So   10:04PM  0:00.05 /hurd/exec
root         5  0.0  0.1  130M  692K  - So   10:04PM  0:00.03 /hurd/auth
root         7  0.0  0.1  130M  868K  - So   10:04PM  0:00.29 /hurd/term /d
root         8  0.0  0.1  130M  684K  - So   10:04PM  0:00.01 /hurd/magic t
root        16  0.0  0.1  130M  732K  - So   10:04PM  0:00.00 /hurd/null
root        17  0.0  0.1  130M  828K  - So   10:04PM  0:00.01 /hurd/storeio
root        36  0.0  0.1  146M  996K  - Sfo  10:04PM  0:00.01 /sbin/syslogd
root        40  0.0  0.1  130M  736K  - So   10:04PM  0:00.01 /hurd/pflocal
root        41  0.0  0.1  146M  928K  - Sfo  10:04PM  0:00.03 /usr/sbin/ine
root        43  0.0  0.2  131M 1.55M  - So   10:04PM  0:00.06 /hurd/pfinet
root        45  0.0  0.1  130M  828K  - So   10:04PM  0:00.01 /hurd/storeio
root        47  0.0  0.1  130M 1012K  - So   10:04PM  0:00.02 /hurd/passwor
root        55  0.0  0.1  145M  688K co S    10:06PM  0:00.01 ps -aux
```

Aquí podemos apreciar todo el tema de servidores y traductores en acción, por ejemplo el primer storeio es el traductor entre el sistema de ficheros ext2 y el dispositivo físico. El siguiente es se encarga de que he montado el cdrom y actúa de traductor entre el dispositivo hd3 y el sistema de ficheros iso9660.

[3]El bash es un intérprete de comandos utilizado en Unix

magic se utiliza para acceder a un dispositivo que no existe fisicamente como una terminal [4].

Como se puede apreciar es un sistema ALTAMENTE INESTABLE, y solo recomendado para gente que le pica la curiosidad o que quiere colaborar en el proyecto.

[4]Como se puede apreciar se encarga de gestionar la única terminal del sistema tty

Capítulo 6

Conclusiones

El sistema Hurd no puede ser sustituto de Linux a corto plazo, dado su actual estado y que muchos programadores y desarrolladores trabajan entorno al núcleo Linux. Seguramente dentro de unos años, el sistema Hurd sea usable a nivel usuario.

En cuanto al soporte de hardware y creación de drivers, como el hardware se encarga de reconocerlo el micro núcleo Mach, no sera una tarea demasiado tediosa el portar los actuales drivers que funcionan en Linux al micro núcleo. El problema esta en como hacer en L4 una serie de primitivas capaces de comunicarse con el micronúcleo para que este acceda al hardware. Implementar esto tal y como se pretende en L4 es una tarea complicada y que aun se esta discutiendo sobre como hacerlo.

También la unidad de gestión de memoria fuera del núcleo, que actualmente en la versión 3 esta dentro, es otro debate de la versión L4. No existen unas posturas claras, sobre como quedará al final la versión L4.

Toda esta inestabilidad ha hecho que la versión Mach 3 haya sido totalmente retirada y no se pueda encontrar ningún mirror en Internet del cual descargarse Debian Hurd con este micronúcleo (al menos en imágenes ISO), a la espera de que algún día no muy lejano se pueda correr el micronúcleo L4.

La verdad merece la pena el esfuerzo de tantos desarrolladores de Hurd, en el sentido de que tal y como esta organizada la fase de creación del núcleo que no depende de una sola persona como es el caso de Linux. Además la posibilidad de acabar el proyecto GNU utilizando el núcleo que tantos problemas ha traído desde su creación.

Mi opinión personal es que le doy la razón a Tanembaum, Linux is obsolete. Ahora mismo Linux funciona relativamente bien, esto es debido a que es abierto y se corrigen muchos fallos, pero en relación a unos años, el kernel ha pasado de tener un compilado óptimo de 400k a tenerlo de 1 mega. No cuesta imaginar que el núcleo seguirá aumentando de tamaño a medida que se le añaden nuevas funcionalidades, dejándose de poder ejecutar en PCs con pocos recursos que era uno de los principios por los que se utiliza Linux. Tampoco recomiendo Hurd a gente que no haya instalado alguna distribución Linux por ejemplo desde cero, como las primeras debian o slackware.

Bibliografía

[Stallings 2001] William Stallings. *Sistemas Operativos 4ª Ed.*, Prentice Hall

[Walfield 2002] Neal H. Walfield Guía de Instalación del Hurd

[Vernon 1999] Matthew Vernon The Easy Guide to Installing Hurd on a Linux Box

[Open 1999] Open Sources: Voices from the Open Source Revolution, Oreilly

[ES HURD 2003] GNU en español

[CMS MACH 2003] Mach Project Home Page

[GNU 2003] Proyecto GNU - Fundación para el software Libre
[1]

[1]Los enlaces web fueron verificados en la fecha de la portada.