Fundamentos de Software de Comunicaciones

Tema 2

Programación del Sistema Operativo

(1ª parte)

Contenidos

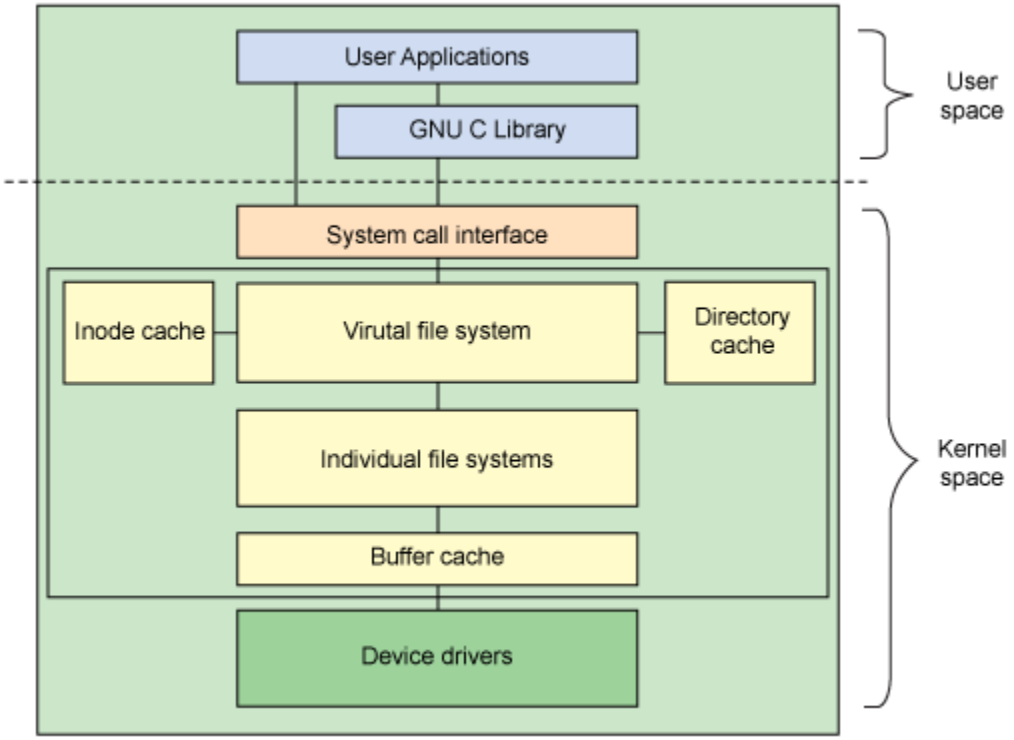
- Introducción
- Funciones de biblioteca vs. llamadas al sistema
- Conceptos básicos de programación de sistemas

Programación de sistemas

- ¿Qué es? Implementar software de sistemas…
- ¿Y qué es el software de sistemas? Es el software de bajo nivel que interactúa directamente con el sistema operativo (ya sea su núcleo –kernel– o librerías del mismo)
- Ejemplo de software de sistemas
  - gedit, gcc, servidor web, una base de datos
- Software de sistemas vs software de aplicación
  - Conocimiento del hardware y el sistema operativo
- Software de comunicaciones como software de sistemas
  - Software desatendido que corre indefinidamente
  - Dan soporte a otros programas/usuarios: un servidor web está usa sockets
  - Requiere mucha eficiencia

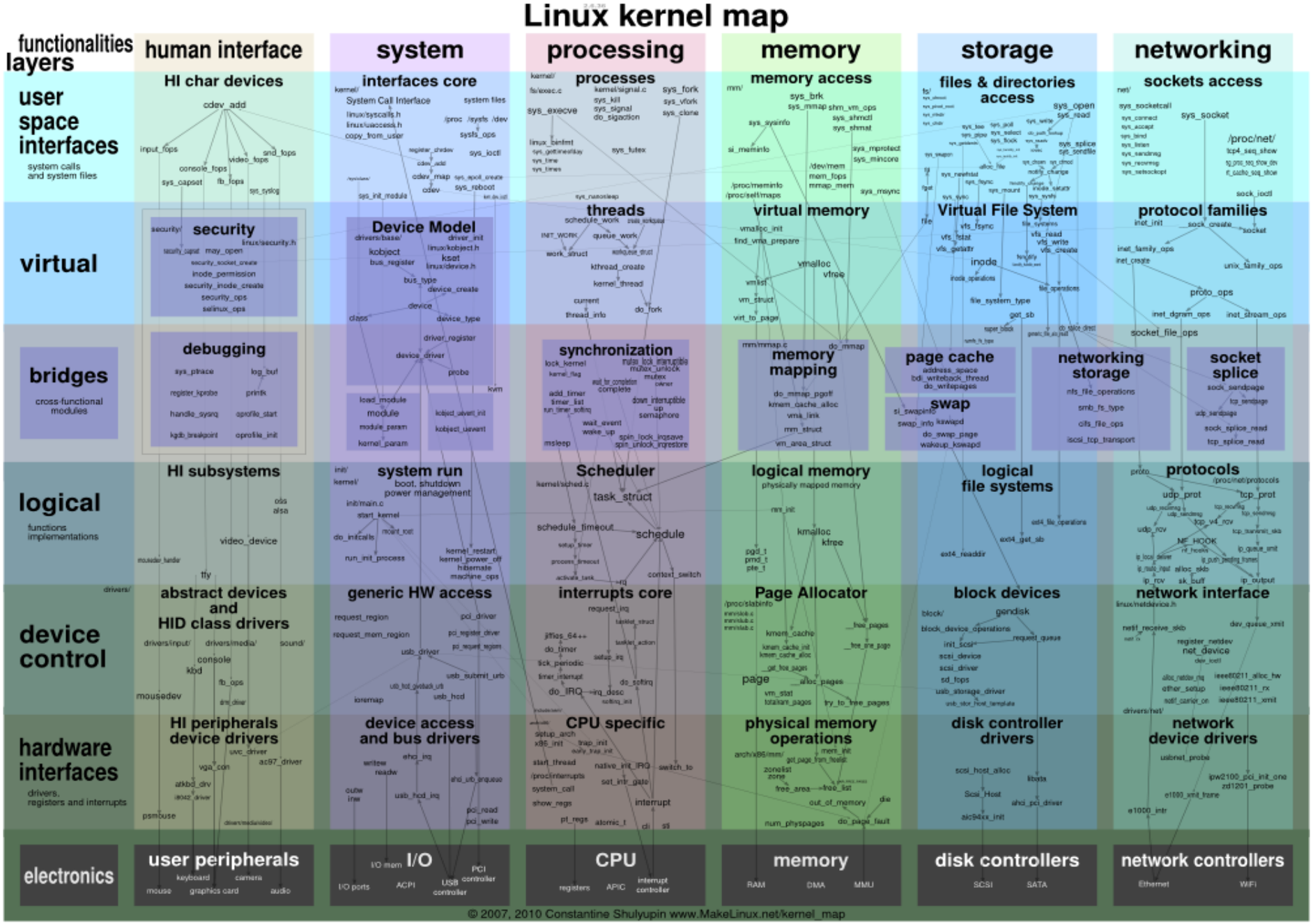
Llamadas al sistema

- Llamada al sistema: invocación a acciones del kernel desde el espacio de usuario para solicitar algún servicio o recurso (e.g., fichero)
- El número de llamadas al sistema depende de la arquitectura (aprox. el 90% de llamadas es común)



Llamadas al sistema (II)

- No se puede vincular (link) código en espacio de usuario con el espacio de kernel
- Por razones de seguridad y fiabilidad no se permite ejecutar directamente código del núcleo desde las aplicaciones de usuario
- La forma de hacerlo es generando una interrupción por software que provoca un cambio de contexto al núcleo
  - la interrupción se maneja con la función *manejadora de llamadas del sistema*
  - *el tipo de llamada al sistema y sus argumentos se recogen de registros (eax, ...)*



Todas las llamadas al sistema aquí: http://www.makelinux.net/kernel_map/intro

Funciones de biblioteca

- Proporcionadas por librerías externas, como GNU C Library
- Su ejecución se realiza en el espacio de direcciones del usuario (a no ser que usen llamadas al sistema ellas mismas → fopen())
- Abstraen las particularidades del sistema concreto y son portables
- Suelen ser más rápidas ya que cambiar de contexto y llevar la aplicación al espacio de direcciones del kernel

Conceptos básicos de la programación de sistemas

- Se recomienda encarecidamente el apartado "**Concepts of Linux Programming**" del libro incluido en la bibliografía:
  - Robert Love, "Linux System Programming", O'Reilly, 2007
  - Disponible en formato electrónico a través de Jábega
    - Edición de 2013: https://www.safaribooksonline.com/library/view/linux-system-programming/9781449341527/?ar&orpq
    - Edición de 2007: https://www.safaribooksonline.com/library/view/linux-system-programming/0596009585/?ar&orpq

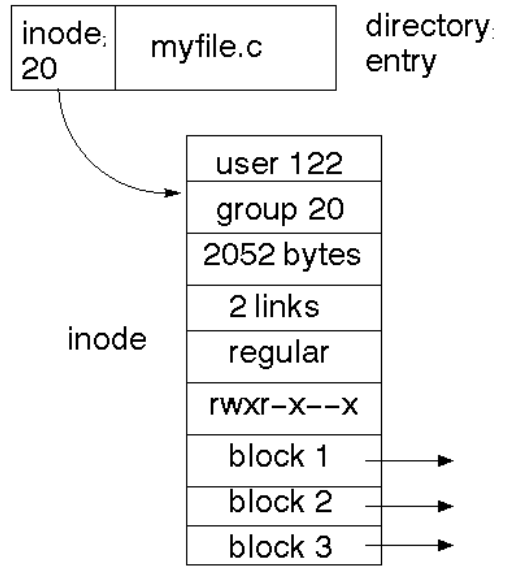
Conceptos básicos de la programación de sistemas

- Ficheros y sistemas de ficheros
  - En Unix/Linux todo es un fichero
  - Abstracción que permite un diseño del software más "homogéneo"
  - La comunicación con dispositivos se realiza a través de lecturas/escrituras a ficheros
  - Trabajo con ficheros
    - Abrir para lectura/escritura/ambos
    - Leer/escribir
    - Cerrar
  - Al abrirlo, se referencia con un descriptor único: el descriptor de fichero (un **`int`** de C)
    - Se comparte entre el kernel y el espacio de usuario

Conceptos básicos de la programación de sistemas

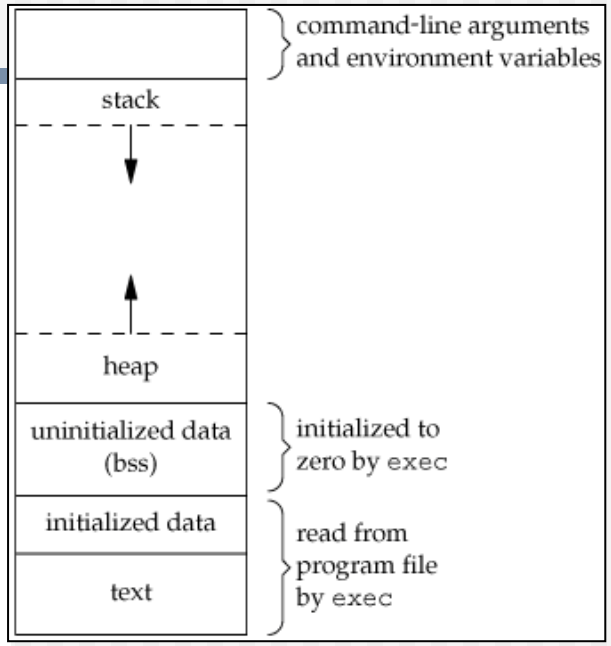
- Tipos:
  - Ficheros regulares: almacenan secuencias de bytes
    - Cabecera, posición u offset: byte por donde se va leyendo/escribiendo
    - Longitud: nº de bytes (truncamiento)
    - Apertura por varios procesos
      - Cada uno, un descriptor
      - No hay control de concurrencia
    - El nombre del fichero no está asociado a sus datos en el sistema de ficheros, si no con un número de i-nodo
  - Directorios
    - Tablas que asocian nombres a i-nodos
  - Enlaces: hard y simbólicos
  - Ficheros especiales

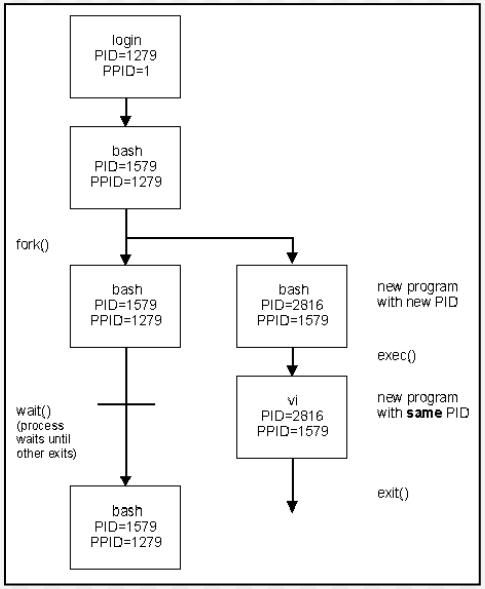
    (representan objetos del kernel)
    - ficheros de dispositivo de caracteres
      - Cola de bytes → Teclado
    - ficheros de dispositivo de bloques
      - Arrays de bytes → almacenamiento
    - tuberías (pipes) con y sin nombre
    - sockets locales (tipo unix)



Conceptos básicos de la programación de sistemas

- Procesos
  - Otra abstracción fundamental en Linux
  - Pieza de código en ejecución, además de
    - Datos (ficheros, temporizadores, señales, etc.), recursos, estado, ordenador virtualizado
  - El núcleo proporciona:
    - un procesador virtual
    - una vista de la memoria también virtual
- Hebras (threads)
  - Actividades dentro de un proceso
- Jerarquía de procesos
  - process id (pid)
  - parent pid (ppid)
- Visualización en consola
  - comando: ps -aux



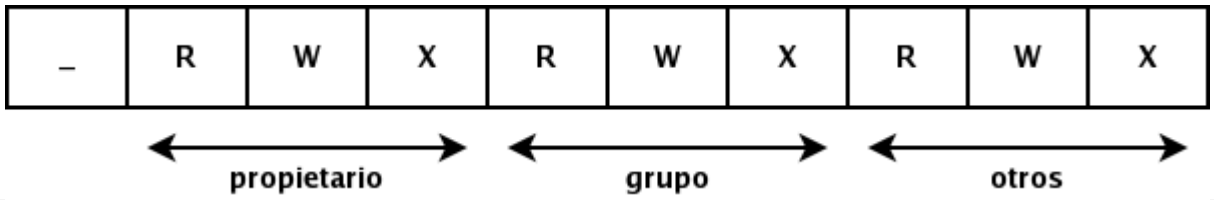


Conceptos básicos de la programación de sistemas

- Usuarios y Grupos
  - El sistema proporciona acceso a ficheros (y procesos) a los propietarios o a los que pertenezcan a un grupo autorizado
    - En el núcleo se identifca a los propietarios con el user id (uid)
    - Creación de usuarios en consola: comando useradd
    - Correspondencia entre nombres de usuarios y su uid en:
      - /etc/passwd

Conceptos básicos de la programación de sistemas

- Permisos en ficheros:
  - R : lectura; W: Escritura; X : Ejecución
  - Primer carácter → tipo de fichero
    - D : directorios
    - - : fichero regular
    - L : enlace simbólico a fichero
    - b y c son tipos especiales de ficheros (bloque/carácter)
  - La forma de ver los permisos de nuestros ficheros es mediante la orden ls -l
  - Cada uno de estos bits pueden activarse o desactivarse juntos o por separado mediante la orden chmod



Conceptos básicos de la programación de sistemas

- Ejemplos de permisos:
  - Si queremos evitar que nadie tenga acceso a un fichero:

chmod 600 secreto.txt

  - Si queremos permitir que nuestro grupo de usuarios pueda modificar un fichero de muestra propiedad:

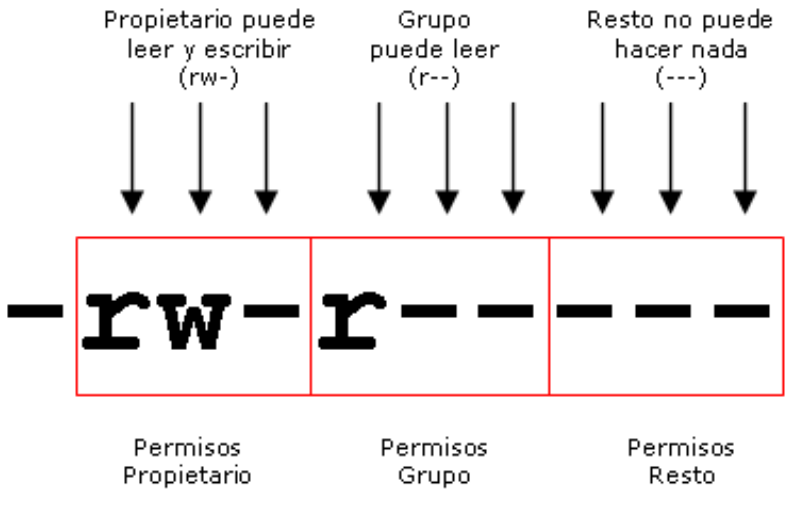
chmod g+rw comun.txt

  - Para permitir la ejecución de un fichero por parte de todos:

chmod o+x instalar.sh

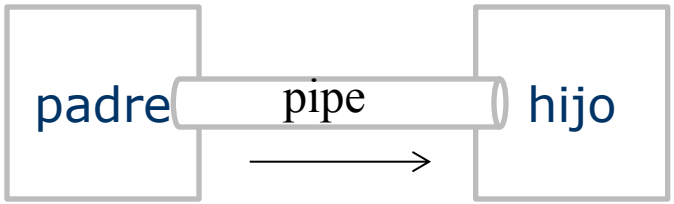
  - Para impedir que otros accedan a un directorio:

chmod o-x dir_secreto



Conceptos básicos de la programación de sistemas

- Mecanismos de comunicaciones
  - Señales: notificaciones asíncronas en un sentido
    - del kernel a un proceso, entre procesos...
    - se atienden en una función manejadora
  - Pipes (tuberías) y pipes con nombre
  - Colas de mensajes
  - Semáforos
  - Memoria compartida
  - Sockets



Conceptos básicos: manejo de errores

- Las llamadas al sistema devuelven -1 cuando sucede un error
- La variable global errno indica el tipo de error que se ha producido

```
if(errno==E2BIG)  cout << "Argument list too long";
if(errno==EACCESS)  cout << "Permission denied";
if(errno== EAGAIN)  cout << "Try again";
...
```

  - En #include <errno.h>

- Función perror(char *) para imprimir el mensaje asociado al error

```
errno = 0;
arg = strtoul (buf, NULL, 0);
if (errno)
    perror ("error en strtoul");
```

- Función char * strerror (int errnum) para devolver el error en formato cadena de caracteres
  - (En #include <string.h>)

Conceptos básicos: el manual de Unix/Linux

- Para consultar todo tipo de información relativa al uso de una llamada al sistema, función de librería C o comando del sistema operativo se utiliza el manual
  - Da información sobre: el prototipo, librerías a incluir, tipo de argumentos, valores de retorno, posibles errores a chequear en *errno,* ejemplos...
- En consola:

  $> man seccion *nombre_funcion_o_comando_unix*

  Ejemplos: *$> man 2 open*

  $> man 3 fopen (igual que $> man fopen)
- El manual se organiza por secciones
  - El argumento sección es opcional, aunque sirve para diferenciar comandos o funciones que se llaman igual y aparecen en distintas secciones

Sección	Descripción
1	Comandos Generales
2	Llamadas al sistema
3	Biblioteca C de funciones
4	Ficheros especiales (normalmente dispositivos, que se pueden encontrar en /dev) y drivers
5	Formatos de fichero y convenciones
6	Juegos y salvapantallas
7	Miscelánea
8	Comandos de administración del sistema y Demonios