Tema 1: Introducción al S.O.

Enrique Soriano

Laboratorio de Sistemas,
Grupo de Sistemas y Comunicaciones,
URJC

15 de febrero de 2010

¿Qué es un sistema operativo?

- Def.- Programas que te dejan usar la máquina, es subjetivo.
- Ventajas:
  - Similar a una biblioteca $\longrightarrow$ reutilización.
  - Abstrae de la máquina: no necesitas conocer los detalles para usarla.
  - Gestiona y reparte la máquina: no necesitas preocuparte de gestionar el tiempo que ejecuta un programa, organizarle la memoria, etc.

¿Qué es un sistema operativo?

- **Máquina virtual o máquina abstracta**: el sistema operativo proporciona una máquina que realmente no existe, es una máquina ficticia que nos ofrece dispositivos virtuales: ficheros, directorios, procesos, conexiones de red, ventanas...
- HW y SO: Compatibilidad hacia atrás. ¿Tus programas dejan de funcionar de una versión a otra del S.O.?

¿Qué es un sistema operativo?

Es un gestor de recursos:

- Multiplexa en tiempo: p. ej. procesador, red.
  - ¿Tienes que preocuparte de soltar el procesador en tu aplicación?
- Multiplexa en espacio: p. ej. memoria, disco.
  - ¿Tienes que preocuparte de no pisar la memoria de otra aplicación?

Kernel

- **Kernel**: núcleo del sistema operativo que gestiona la memoria, el procesador, **drivers** para dispositivos (USB, PCI, VGA...), protocolos de red, sistemas de ficheros, etc. Ejecuta en modo privilegiado (modo kernel o supervisor): puede ejecutar cualquier instrucción de la CPU, acceder a toda la memoria, etc.
- **Aplicaciones o procesos de usuario**: ejecutan en modo no-privilegiado, tienen restricciones. Piden servicio al **kernel** cuando necesitan algo que no pueden hacer mediante **llamadas al sistema**.

Tipos de kernel

- **Kernel monolítico**: El kernel es un único programa. Puede tener carga dinámica de módulos.
- **Microkernel**: El kernel proporciona servicios básicos (gestión de memoria, gestión del procesos, gest etc.) y los otros servicios ejecutan como procesos de usuario llamados *servidores*, comunicándose con el microkernel mediante mensajes.
- **Híbridos**: Diseño basado en microkernel, implementado como monolítico.

Definiciones

- **Terminal**: máquina que te permite ejecutar comandos, compuesta hoy en día por display, teclado y ratón como dispositivos de I/O.
- **Comando o mandato** (command): cadena de texto que identifica a un programa u orden.
- **Shell**: programa que te deja ejecutar *comandos*. Por lo general, permite crear programas (scripts) en un lenguaje propio. En Plan 9, `Rc`.
- **Prompt**: texto que indica que el **shell** está esperando una orden.

Entorno: Plan 9

- **Servidor de ficheros**: máquina que contiene los ficheros del sistema y de los usuarios.
  - Los programas que ejecutan en el terminal son sus **clientes**.
  - Los terminales no mantienen estado. El estado de los ficheros se mantiene en el servidor de ficheros.

Nota: Cuando instalamos Plan 9 en local, el servidor de ficheros ejecuta en el terminal, por tanto sí mantendrá estado.

Entorno: arranque

- ¿Cómo se arranca un sistema? Simplificando:
- En local:

  BIOS → Cargador (disco, MBR) → Cargador (disco, partición x) → Kernel (disco, partición x)
- De red:

  BIOS → BIOS PXE (tarjeta de red) → configura red (DHCP) → Cargador (por TFTP) → Kernel (por TFTP)
- Una vez arrancado el kernel, necesitas un sistema de ficheros. Puede requerir **autenticación**: nombre de usuario y contraseña. El agente de autenticación en Plan 9 es `Factotum`.

Entorno: GUI

- `Rio` es el manejador de ventanas (*window manager*): crear ventanas, matar ventanas, redibujarlas, esconderlas...
- `Acme` es un editor de texto para programadores, un navegador de ficheros, y un shell. Tres en uno.

Comandos básicos

- **Carácter de escape**: significa para Shell otra cosa que no es el propio símbolo que representa. P.ej. backslash (barra invertida), comillas simples, etc.
- Los comandos pueden aceptar argumentos.
- Los argumentos pueden ser modificadores (-a, -l, -s ...), que cambian el comportamiento del comando.
- MAYÚSCULAS $\neq$ minúsculas.

Comandos básicos

- `date`
- `touch`
- `ls`
- `lc`
- `cp`
- `mv`
- `rm`

Directorios

- Def - Agrupación de ficheros y directorios. Son ficheros.
- Dos ficheros que están en distintos directorios son dos ficheros diferentes.
- Organizados en **árbol** → directorio **raíz** (root).
- **path absoluto** vs. **path relativo**.
- **Directorio de trabajo**.
- . (dot), .. (dot-dot), $home, cd , pwd.

Directorios

```
term% mv f1 f2

term% mkdir /a/dir ; mv /a/dir /b
            # error! mv sólo renombra directorios

term% ls -s dir ; ls -sd dir

term% cp f1 f2 f3 /dir

term% cp a /dir ; rm /dir
            # error! un directorio con contenido no se puede borrar!

term% rm -rf /dir
            # mucho cuidado con  -r!
```

Ficheros y datos

- **Fichero**: secuencia de bytes, con un cierto nombre.
- **Extensión**: sólo es parte del nombre.

Ficheros y datos

- `cat`: concatena y muestra el contenido de un fichero.
- `xd`: muestra el contenido de un fichero en distintos formatos.
  - -b separa por bytes
  - -c imprime el carácter ASCII o la representación del código de escape si no es imprimible.
  - -b -c , ambas cosas a la vez.
- `file`: te da pistas sobre el contenido de un fichero.

Representación de texto plano

- ASCII: usa 1 byte para almacenar caracteres (7 bits en realidad).
- ISO-Latin 1 (8859-1): usa 1 byte para almacenar caracteres (8 bits).
- UTF-8: puede usar 1,2, o más bytes. Compatible hacia atrás. Plan 9 usa esta codificación.

El comando `tcs` sirve para traducir codificaciones de texto.

Caracteres de control

- El carácter `'\n'` indica nueva línea en el texto. Es una convención usada en todos los programas, bibliotecas, etc. en Plan 9 (también en otros sistemas UNIX).
- En otros sistemas operativos no tiene por qué ser así (Windows usa la secuencia `'\n\r'`).
- El carácter `'\t'` indica un tabulador.
- No hay carácter EOF: invención de los lenguajes.
- Ctr+d es EOT: *fin de transmisión* (carácter 0x04).

Permisos

- Cada fichero tiene un dueño y un grupo.
- Lista de permisos para el dueño, el grupo y los demás usuarios.
- El modo indica: tipo (fichero, directorio, fichero *append only*), si es de apertura exclusiva, y los permisos para leer, escribir y ejecurar (RWX) para el dueño, el grupo y los demás.

```
[da-] [l-] [r-][w-][x-] [r-][w-][x-] [r-][w-][x-]
```

Permisos

- En Plan 9 los permisos se aplican en cascada, de menos a más privilegiado.
- En el servidor de ficheros hay al menos un grupo por cada usuario, y podemos crear más (p. ej. ssoo).

Permisos

Ficheros:

- R: se pueden leer los bytes del fichero.
- W: se pueden modificar los bytes del fichero, o truncarlo.
- X: se puede ejecutar.

Permisos

Directorios:

- R: se pueden listar las entradas del directorio.
- W: se puede crear/borrar/modificar entradas de directorio (crear un fichero, cambiar su nombre, etc.).
  Nota: si el directorio tiene el permiso W, se puede borrar un fichero independientemente de sus permisos.
- X: se puede buscar en el directorio.
  Nota: aunque el directorio no tenga el permiso R, se puede acceder a un fichero/directorio si se conoce de antemano su nombre.

Para más detalles, *stat(2)*.

Permisos (sistema de ficheros)

Creación:

- Dueño: usuario que lo crea.
- Grupo: grupo del directorio en el que se crea.
- Permisos:
  si es fichero:
  ```
  perm & ((~  0666) | (0666 & dir.perm))
  ```
  en la shell y touch, perm=0666

  si es directorio:
  ```
  perm & ((~  0777) | (0777 & dir.perm))
  ```
  en mkdir, perm=0777