# Sistemas Operativos

# Practica de laboratorio

# Funciones del Sistema Operativo

Daniel Ochoa

dochoa@fiec.espol.edu.ec

### Objetivos:

* Demostrar las funciones del Sistema Operativo que fueron aprendidas en clase.
* Identificar la estructura de los directorios de un sistema Operativo.
* Identificar la estructura de un Sistema Operativo basado en UNIX.
* Realizar tareas de monitoreo de recursos empleando un shell.
* Encontrar llamadas a sistemas, APIs y sus funciones en un proceso.
* Identificar los procesos iniciales de un sistema.

### Requerimientos:

* Un computador con GNU/Linux.
* Versión de Kernel >= 2.6.24

### Nota:

* Escriba **man <nombre del comando>** para conocer como funciona un comando.

### Actividad No 1: El sistema

Los sistema operativos dan soporte a ciertas plataformas de hardware, una vez instalado hay como saber en que tipo de plataforma se está trabajando y la version del kernel que se utiliza. Con el comando **uname -a** deduzca lo siguiente:

|  |  |  |
| --- | --- | --- |
| **A** | 1. nombre kernel | Linux |
| 2. version de kernel | #1 SMP Wed Feb 19 06:37:35 UTC 2020 |
| 3. procesador | x86\_64 |
| 4. Arquitectura de máquina | x86\_64 |

### Actividad No 2: Componentes del Kernel

Linux es un kernel modular, es posible saber que módulos estan cargados en memoria, el archivo correspondiente y sus parámetros. Utilice los comandos **lsmod y modinfo** para buscar un módulo relacionado al manejo del procesador y escriba su archivo de módulo (\*.ko) abajo

|  |  |  |
| --- | --- | --- |
| **B** | 1. nombre módulo | acpi\_pad |
| 2. archivo de módulo | /lib/modules/5.6.0-1028-oem/kernel/drivers/acpi /acpi\_pad.ko |

### Actividad No 3: Uso de recursos del sistema

Se pueden conocer el uso de los recursos del sistema usando varios programas. Algunos ejemplos están abajo, explica que se obtiene al ejecutarlos:

|  |  |  |
| --- | --- | --- |
| **C** | 1. df | Muestra todas el espacio usado y disponible de todas las unidades montadas en el sistema |
| 2. free | Muestra la espacio total, usado, libre, compartido y cacheado de la memoria principal |

Como en UNIX todo es un archivo es posible leer archivos de dispositivo para ver el estado del mismo. El CPU se representa con el archivo **/proc/cpuinfo.** Lea este archico con el comando **more** y anote el número de CPU cores y su velocidad.

|  |  |
| --- | --- |
| **D** | Son 8 CPUs  0: 3250 MHz  1: 3603 MHz  2: 3600 MHz  3: 3443 MHz  4: 3659 MHz  5: 3353 MHz  6: 3553 MHz  7: 3127 MHz |

Un comando útil es **top.** Liste 2 ejemplos de tipos de recursos se pueden ver en uso? Para terminar la ejecución de **top**, presione las teclas ctrl + C, ó ctrl + Z .

|  |  |
| --- | --- |
| **E** | 1. Porcentaje del uso CPU |
|  | 2. Porcentaje del uso de la memoria principal |

Antes de salir de **top** pulse 1, qué observa?

|  |  |
| --- | --- |
| **F** | Se muestran estadísticas por cada core del CPU |

El uso de la red también puede ser monitoreado. Con el comando **netstat -i** extraiga los siguientes parámetros de alguna interfaz de red (por ejemplo eth0, tarjeta ethernet):

|  |  |  |
| --- | --- | --- |
| **G** | 1. paquetes enviados | 3208571 |
| 2. paquetes recibidos | 2534540 |

### Actividad No 4: Excepciones.

Las excepciones son interrupciones activadas por hardware cuando se recibe una señal eléctrica desde el dispositivo. El comando **lsusb** lista dispositivos con interfaz usb. Observe la información que brinda el comando. Ahora conecte un dispositivo usb, vuelva a escribir el comando. ¿Pudo observar algún cambio?

|  |  |
| --- | --- |
| **H** | Cuando se conecta el dispositivo USB y se ejecuta el comando, se muestra el nuevo dispositivo conectado |

Ahora ejecute **lspci** que tipo de información que proporciona la salida de este comando.

|  |  |
| --- | --- |
| **I** | Muestra todos los dispositivos conectados a través de la interfaz PCI |

### Actividad No 5: Programas

El sistema de archivos proporciona la forma de almacenar programas y datos a través de una organización jerárquica basado en directorios. El directorio / se denomina directorio raíz. A partir de este se ubican los demás directorios, entre los cuales se encuentran:

|  |  |
| --- | --- |
| /home | Datos de usuarios |
| /dev | Dispositivos del sistema |
| /var/log | Mensajes generados por procesos en background |
| /usr/src/linux..../kernel | Código fuente kernel |

Para acceder a cada directorio, se usa el comando **cd** <nombre directorio>. Con **cd ..** se regresa al directorio superior. Acceda a al directorio **/bin** y observe su contenido usando el comando **ls.** Reconoce a que corresponden los archivos en ese directorio?

|  |  |
| --- | --- |
| **J** | Son los archivos de los programas del sistema y también programas que el usuario ha instalado |

### Actividad No 6: Procesos

**Top** también provee de un resumen del estado de los procesos. Indique abajo que proceso consume la mayor cantidad de memoria en su sistema

|  |  |
| --- | --- |
| **K** | Docker |

Adicionalmente, se puede ver la lista de procesos cargados en memoria usando el comando **ps** y sus opciones, prueba las combinaciones descritas abajo, que diferencias hay entre ellas?

|  |  |  |
| --- | --- | --- |
| **L** | 1. ps -ax | Muestra todos los procesos del sistema con la sintaxis BSD |
| 2. ps -axjf | Muestra todos los procesos y sus relaciones padre-hijo en una estructura de árbol |
| 3. ps -u root | Muestra todos los procesos que sean del usuario root |

Usando estos resultados es posible deducir que programa terminal se está usando. Anótelo a en la casilla de abajo.

|  |  |
| --- | --- |
| **M** |  |

El kernel eventos en un ring buffer, este buffer puede ser leido para saber que ocurre en la computadora usando el comando **dmesg.** Si ejecuta **dmesg -c** el buffer se borra. Borre el buffer y conecte un dispositivo usb, si ejecuta de nuevo **dmesg** que obtiene?

|  |  |
| --- | --- |
| **N** | Se pueden ver todos los eventos que ocurriendo cuando se conectó el nuevo dispositivo usb |

### Actividad No 7: Interrupciones en software

Es posible enviar interrupciones en software (señales) entre procesos, una manera de hacer esto es usando el comando **kill** (número), dependiendo de el parámetro número diferentes cosas pueden pasar. Pruebe lo siguiente: 1) ejecute el programa **gedit** (editor de texto de Gnome). 2) En un terminal ejecute **ps -ax** y extraiga el id del proceso geditluego ejecute cada uno de los siguientes comandos y describa que pasa con **gedit.**

|  |  |  |
| --- | --- | --- |
| **O** | 1. kill -19 gedit\_id | El proceso se suspende |
| 2. kill -18 gedit\_id | El proceso continua |
| 3. kill -9 gedit\_id | El proceso de termina |

Más información **kill -l**

### Actividad No 8: Llamadas al sistema

Los procesos emplean los servicios del sistema operativo por medio de llamadas al sistema. Una forma de conocer las llamadas al sistema que se hacen es utilizando el comando **strace** seguido por el nombre del programa a ejecutar. Ejecute **strace date** y anote 4 llamadas al sistema:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | 1.openat | 2.mmap | 3.read | 4.write |

En la mayoría de los casos los procesos hacen llamadas al sistemas a travez de librerias de programación. Con el comando **ldd** se puede conocer de que librerías de las que depende un programa. La forma de usarlo es la siguiente: **ldd** <path programa>. Escriba las dependencias del programa **/bin/cat**.

|  |  |
| --- | --- |
| **Q** | Linux-vdso.so.1  Libc.so.6  /lib64/ld-linux-x86-64.so |

Para ver las funciones que provee un API se usa el comando **nm -D** <directorio\_librería/ nombre\_librería>. Busque en que librería utilizada por el programa **cat** esta implementada la funcion **printf.**

|  |  |
| --- | --- |
| **R** | En Libc.so.6 |

*Sugerencia: después del comando nm usa “****| grep printf****”*

### Actividad No 9: Configuración del sistema.

Una vez que el kernel se encuentra en memoria, este carga varios procesos de forma automática. Los directorios **/etc/rcX** contiene scripts para estos programas. Estos cambian según X,el runlevel (nivel de ejecución) del sistema. Existen hasta 7 niveles numerados. Describa que configuración tiene el sistema para los runlevels del 1 al 5

|  |  |
| --- | --- |
| **S** | 0. Runlevel 0: Detener sistema  1. Runlevel 1: modo de un solo usuario (para administración)  2. Runlevel 2: modo multiusuario local (sin NFS, network file system)  3. Runlevel 3: modo multiusuario y también NFS (network file System)  4. Runlevel 4: no es usado  5. Runlevel 5: modo multiusuario, NFS e interfaz gráfica. |