Trabajo Práctico 1

Práctica con Linux

Para las prácticas en Linux, hemos elegido la distribución Ubuntu. Se trata de un sistema GNU/Linux instalado en CD-ROM también llamado LiveCD. Es decir que no es necesario que se instale en el disco rígido de la computadora.

Esto facilita enormemente las prácticas, ya que instalarlo en el disco rígido requiere mayor conocimiento acerca del uso de este sistema y esto se encuentra fuera del alcance de la materia.

Existen varias distribuciones de GNU/Linux de este estilo (Live CD, en inglés o Disco Compacto Vivo) y ésta es, tal vez, la más conocida y divulgada.

Además podemos agregar que tiene una muy buena detección del hardware, lo que la hace muy adecuada para utilizarla en una amplia gama de computadoras personales.

Cabe aclarar ahora que las versiones Live CD de Ubuntu pueden estar instaladas tanto en un CD como en un DVD o en una memoria con conector USB (Pen Drive).

Nos referiremos genéricamente a él como CD sabiendo que puede estar en cualquiera de los otros dos soportes.

Descarga y comprobación

Supongamos que vamos a utilizar Ubuntu en alguna de sus variantes. Ubuntu produce dos versiones por año: una en abril y la otra en octubre, y los números de la versión se refieren al año y mes de su aparición. Entonces, la versión 22.04 corresponde a la del año **2022**, mes **04**, es decir, abril de 2022. Las versiones indicadas como LTS (*Long Term Support*) tienen soporte para cinco años (la versión 22.04 es LTS), y es la que utilizaremos.

Además de la distribución Ubuntu «oficial», que emplea el entorno de escritorio GNOME, contamos con algunos derivados o variantes (también se los suele llamar «sabores»), entre ellos:

- Kubuntu: Es una distribución Linux que utiliza KDE como entorno de escritorio.
- Lubuntu: Es una distribución Linux ligera que emplea el entorno de escritorio
- Xubuntu: Es una distribución Linux basada en Ubuntu que usa el entorno de escritorio Xfce.

- Ubuntu Budgie: es una distribución oficial del proyecto Ubuntu que utiliza el entorno de escritorio Budgie, basado en GNOME.
- Ubuntu MATE: Es una distribución Linux basada en Ubuntu. Está mantenida por la Comunidad y es un derivado de Ubuntu, oficialmente reconocido por Canonical, usando el entorno de escritorio MATE, derivado del código base de GNOME 2.
- Ubuntu Studio: Es una distribución Linux basada en Ubuntu. Está orientada a la edición multimedia profesional de audio, video y gráficos.
- Mythbuntu: Es una distribución Linux basada en Ubuntu y el software MythTV, que es un conjunto de aplicaciones que intenta convertir el PC en una grabadora de video digital, con el hardware adecuado.

Cualquiera de éstas puede descargarse a partir de:

http://cdimage.ubuntu.com/

Entonces, puede descargar la versión más liviana Lubuntu desde

http://cdimage.ubuntu.com/lubuntu/releases/22.04/release/

También puede utilizar Ubuntu con escritorio KDE, que es también bastante liviano:

http://cdimage.ubuntu.com/kubuntu/releases/22.04/release/

Hay versiones para 64 y 32 bits, de acuerdo con la arquitectura de su equipo. La mayoría de las computadoras actuales están basadas en procesadores de 64 bits pero las que aparecieron hace unos años suelen ser de 32 bits.

Inicialmente utilizaremos la versión «Desktop» o de escritorio, luego de unas clases más probaremos la versión «Server».

Es muy fácil grabar la imagen ISO en un CD, DVD o pendrive desde Windows.

Una vez finalizada la descarga es una buena idea comprobarla. Esto se hace mediante la «suma de comprobación» o *checksum*. En las páginas de descarga puede encontrar un archivo que se llame, por ejemplo, MD5SUMS o SHA256SUMS, que contienen los números de *checksum* según distintos algoritmos. El algoritmo MD5 realiza una reducción criptográfica de t28 bits y el SHA uno de 256. Es decir, sin importar el tamaño del archivo original, el algoritmo calcula mediante una función *hash* un número de 128 o 256 bits. Si el archivo descargado hubiera sufrido una alteración durante la descarga, este número será distinto al original calculado y publicado en el sitio de origen. En este caso, NO debemos utilizar la imagen ISO descargada.

¿Cómo calculamos este número en Windows? Mediante el programa QuickHash, con licencia libre GPL y disponible en https://www.quickhash-gui.org/

Por ejemplo, el número SHA256SUM del archivo **lubuntu-22.04.1-desktop-amd64.iso** es:

55fa19c398c23d7721755aa1f4738ebbf03613eeba9fc33aa98cbc324b1e03b1

Sólo si obtenemos este número podemos proceder a la grabación de la imagen ISO o a su utilización en una máquina virtual.

Grabación de la imagen en Windows

Es muy fácil grabar la imagen ISO en un CD, DVD o pendrive desde Windows. Existen muchos programas para hacerlo pero uno de los más recomendados es Rufus, con licencia libre GPL y descargable desde

https://rufus.akeo.ie/

No obstante, si usted ya tiene una computadora con alguna distribución GNU/Linux instalada, úsela sin dudarlo, ya que la desventaja que presenta esta distribución es que utiliza la memoria RAM como si fuera un disco rígido; por lo tanto, los cambios a los archivos de configuración y la creación de nuevos archivos son volátiles, es decir que no permanecerán después de un reinicio de la computadora.

Por otra parte, si no tiene experiencia en el uso y administración de un sistema GNU/- Linux, con esta distribución corre pocos riesgos de arruinar su disco rígido o el sistema operativo que tenga en él (supuestamente Windows).

Iniciar el Live CD

Ubuntu es un CD-ROM con arranque (*boot*), es decir que para cargarlo debe asegurarse de que su computadora es capaz de arrancar un sistema operativo desde la unidad de CD-ROM. Verifique esto en el *setup* de la BIOS y establezca como primer dispositivo de arranque a la unidad de CD-ROM.

Al arrancar Ubuntu verá en la primera pantalla (Figura 2) las opciones (en inglés) «Probar Ubuntu sin instalarlo», «Instalar Ubuntu», «Revisar si el disco tiene defectos», «Examinar la memoria», «Arrancar desde el primer disco rígido». En la parte baja de la pantalla tenemos las opciones para la elección del lenguaje, que por omisión está en inglés. Tiene a su disposición una ayuda presionando la tecla Ft; con la tecla F2 vuelve a la pantalla de elección del lenguaje; con la tecla FL puede cambiar la distribución del teclado -que por omisión es la de Estados Unidos (US)-; con la tecla F4 puede cambiar los «modos» de arranque de «Normal» a «Modo gráfico seguro», en el que hace menos detección automática de hardware, «Use driver update CD2, «OEM install (for manufacturers)»

Figura 2: Pantalla de arranque de Ubuntu

Entonces cambiaremos el idioma a «Español» tocando la tecla F2 y veremos la Figura 3

y tal vez deba cambiar la distribución de teclado de «Español» a «Latinoamericano» oprimiendo la tecla F1. Quizá no sepa cuál es la diferencia: Una de ellas es que en el teclado «Español» el símbolo «@» está en la misma tecla que el número «2», en cambio en el teclado «Latinoamericano» lo tiene en la misma tecla que la letra «Q».

Con la tecla F5 tiene opciones para lograr mejor accesibilidad. Estas opciones son: «Ninguna», «Alto contraste», «Lupa», «Lector de pantalla», «Terminal Braille», «Modificadores de teclado» o «Teclado en pantalla». Con la tecla F6 tiene otras opciones que son las que se le pueden pasar como parámetro al núcleo.

Al arrancar Ubuntu en «Modo Normal», aparece el logo de Ubuntu y una barra de desplazamiento mientras lee el CD; puede tardar unos minutos, dependiendo de la velocidad del lector de CD y la de su procesador.

Tenga en cuenta que al ejecutarse desde el CD-ROM, que es mucho más lento que un disco rígido, muchas veces tendrá la sensación de que el sistema está «congelado» o «colgado» (como se dice en la jerga) y no es así. Por lo tanto, tenga paciencia y no comience a sacudir el ratón frenéticamente ni apretar teclas, tratando de obtener alguna respuesta.

Si lo intranquiliza esperar que termine de arrancar y prefiere ver qué está pasando (aunque en esta etapa no entienda bien lo que es) puede modificar los parámetros que se le pasan al núcleo durante el arranque presionando la tecla F6, como vimos. Ahí verá en la línea de parámetros los que dicen «quiet splash». El primero provoca que no aparezcan mensajes provenientes del núcleo durante su inicio. El segundo provoca que aparezca la

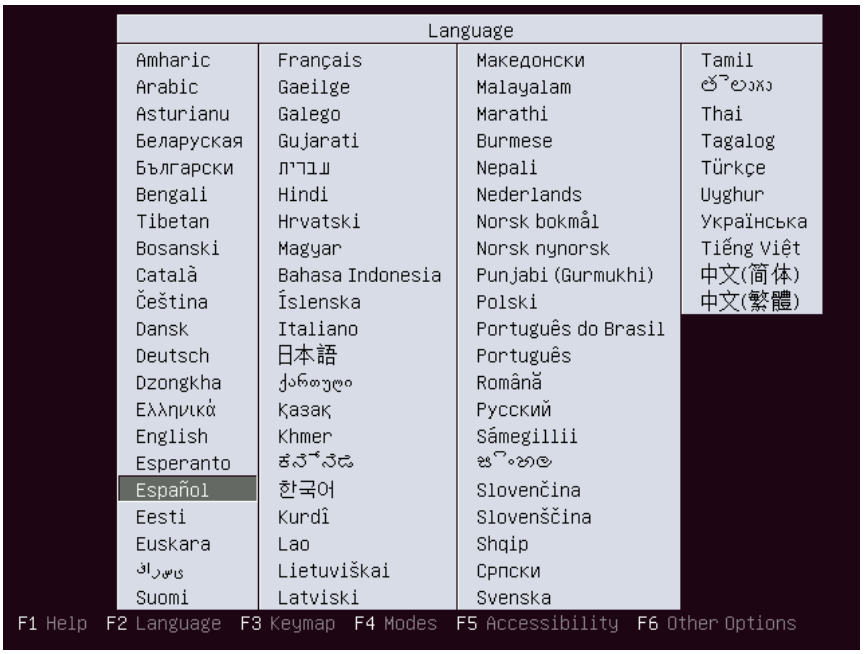

Figura 3.: Cambiando el idioma

pantalla gráfica con su barra de progreso, y ésta oculta cualquier otro mensaje de texto que pudiera aparecer por detrás de ella.

Entonces borre cuidadosamente esas dos palabras y presione la tecla «Enter» para comenzar el arranque de Ubuntu.

Ingresar al sistema

Finalmente aparece la pantalla de ingreso (*login*), en la que le pide ingresar el nombre de algún usuario; si no lo hace en unos pocos segundos, ingresa como usuario ubuntu, que es la opción por omisión y se trata de un usuario sin privilegios administrativos en el sistema. Si funciona el ingreso automático, verá el Escritorio de Ubuntu, como se muestra en la Figura 4:

2.6.S.C. Sesión gráfica

En general, el escritorio de trabajo está compuesto por:

- El botón de Inicio (*Dash*, en inglés).
- El Tablero del sistema.
- El lanzador de aplicaciones (*Launcher*).
- La barra superior de opciones.

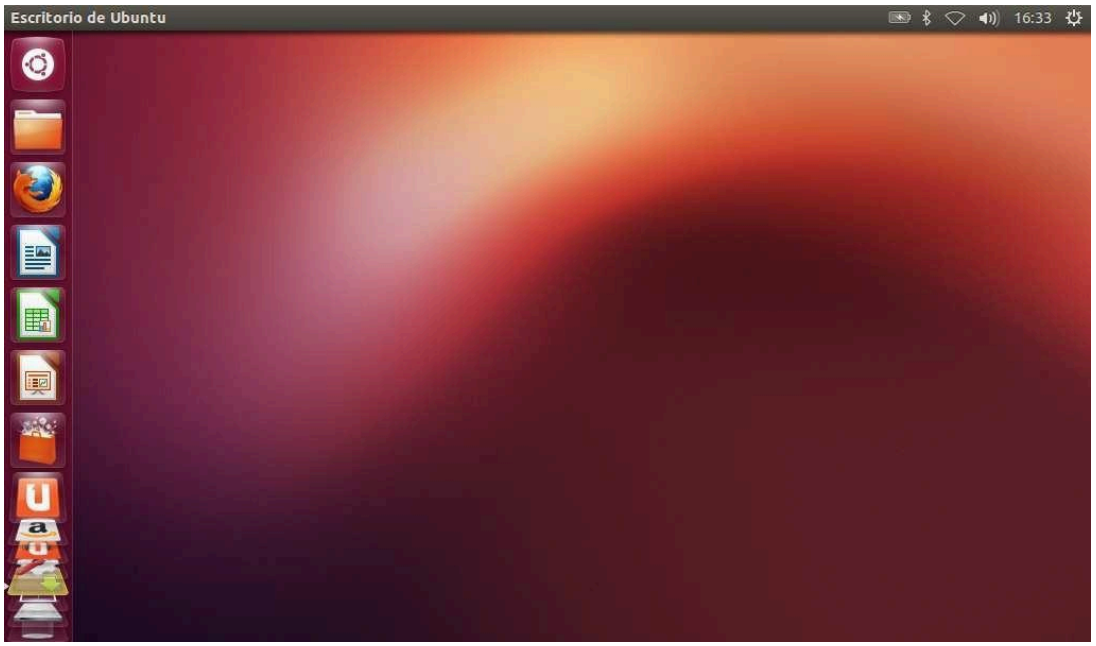

Figura 4.: Unity - Escritorio de Ubuntu

En la parte superior izquierda de su Escritorio tiene el botón de Inicio (Figura 5):

Permite lanzar el Tablero del sistema, que vemos en la Figura 6, para interactuar rápidamente con las búsquedas de archivos y principales aplicaciones.

Figura 5.: Botón de Inicio

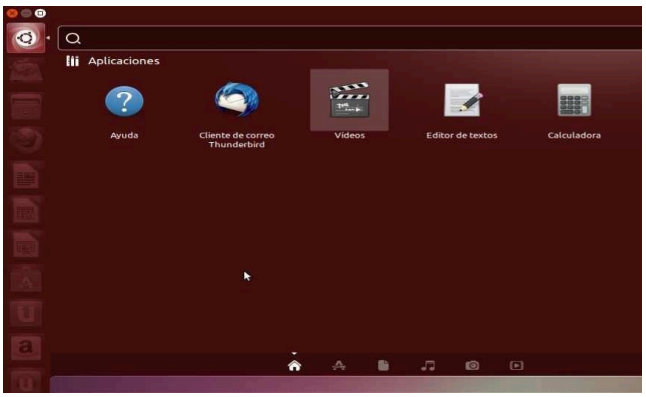

Figura 6.: Tablero del sistema

Línea de comandos

La gran mayoría de las distribuciones Linux proveen varias terminales virtuales o «consolas», que logran imitar a las reales gracias a una combinación de teclas. Por tratarse Linux de un sistema operativo multiusuario, es posible que más de un usuario esté usando el sistema. Se provee de varias

consolas desde las cuales puede iniciar sesión; presionando Ctrl+Alt+F1 ... F6 puede permutar entre las seis consolas e ingresar con distintos usuarios (o el mismo si lo desea, pero las sesiones serán tratadas como distintas). En el caso particular de Ubuntu, todas estas consolas ya están iniciadas con el usuario ubuntu. Luego, si presiona la combinación Ctrl+Alt+F7, retorna a la pantalla gráfica.

Necesitaremos tener acceso a una CLI, pero no es estrictamente necesario cambiarnos a las terminales virtuales. Podemos tener acceso a una de ellas dentro de una ventana gráfica.

Si en la caja de búsqueda del Tablero comenzamos a escribir «term» nos aparecerán las distintas aplicaciones de terminal para líneas de comandos (CLI), como se muestra en la Figura 7:

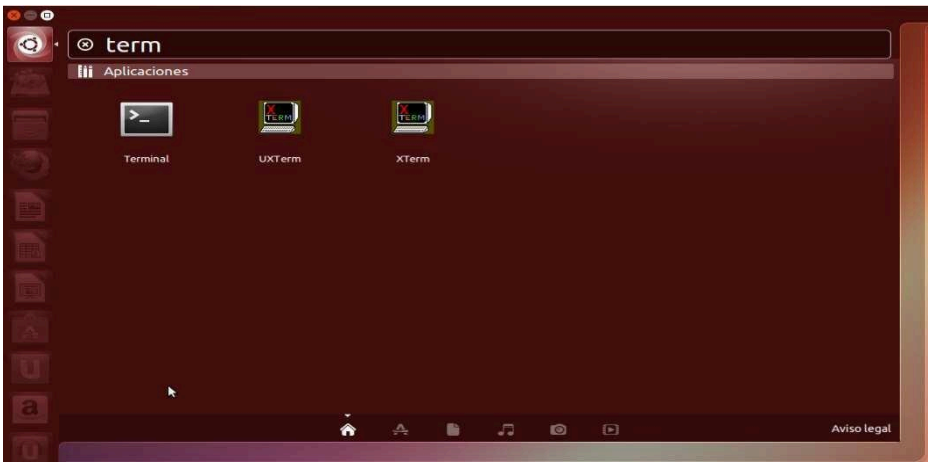

Figura 7: Aplicaciones de terminal para línea de comandos - CLI

Utilizaremos la primera «Terminal» que corresponde a la aplicación denominada «gnome-terminal», que vemos en la Figura 8:

También podemos tomar un «atajo» para lanzar esta aplicación si presionamos simultáneamente la combinación de teclas «Ctrl+Alt+t».

Ahora que usted está ejecutando esta aplicación, está haciendo uso de una interfaz de línea de comandos (o *Command Line Interface*, *CLI*, en inglés). El intérprete de comandos utilizado es el bash (*Bourne Again Shell*).

A la línea que dice «ubuntu@ubuntu:~$» se la denomina *prompt*, en inglés. Es un símbolo o indicación de espera a que usted introduzca algún comando. Lo que usted introduzca, luego de presionar la tecla Enter, intentará ser interpretado y, si es posible, ejecutado por bash.

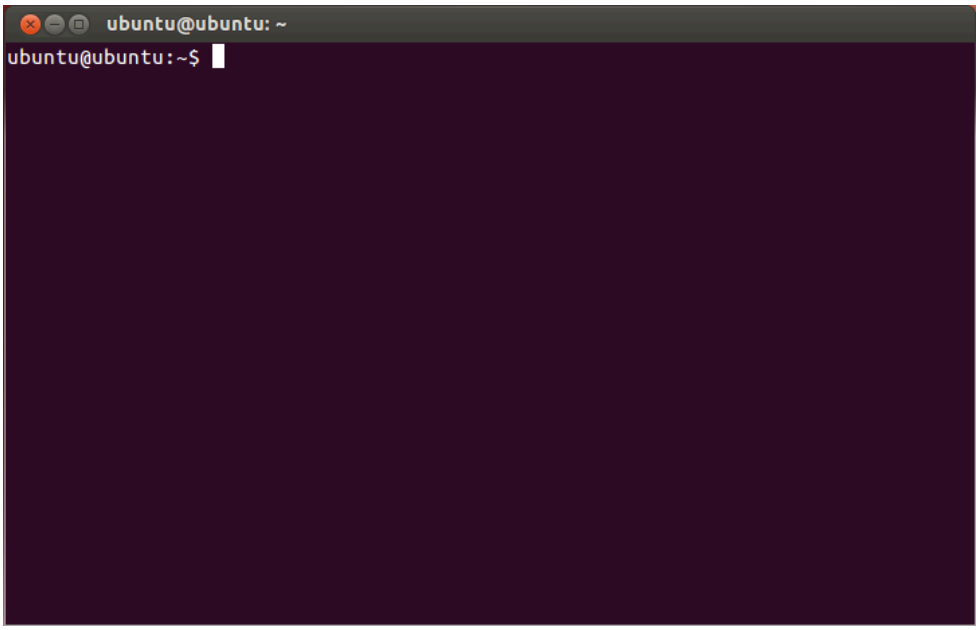

Figura 8: Gnome Terminal de Ubuntu

El *prompt* es muy personalizable y puede tomar muchas formas. Ésta en particular significa «usuario en máquina», es decir, «usuario ubuntu en máquina llamada ubuntu». El símbolo arroba (@) es llamado «*at sign*» en inglés, es decir signo «en». Luego, el símbolo «~» significa que estamos posicionados en nuestro directorio personal o «directorio casa» (*home directory*, en inglés). Finalmente, el símbolo «$» significa que estamos usando un usuario sin privilegios administrativos. En caso de estar utilizando un usuario con privilegios administrativos, el símbolo será «#».

Cierre de sesión y apagado

Si usted escribe como comando a «exit» cerrará su sesión de línea de comandos y se cerrará la ventana "Terminal", pero permanece en su sesión gráfica.

Arriba y a la derecha del escritorio tiene el botón de apagado.

Que le permitirá apagar o volver a iniciar su computadora. En cualquier caso, el CD o DVD será expulsado para que usted lo saque, cierre la compuerta y presione la tecla «Enter» para finalizar con el apagado. De esta manera, la próxima vez que la encienda se hará un arranque normal del sistema que tenía instalado en su disco rígido.

Comandos básicos

Ya sea que no haya apagado su sesión anterior o que haya decidido hacerlo e iniciar todo desde el principio, ahora retomaremos nuestra sesión con el intérprete *bash*.

Decíamos que estábamos posicionados en nuestro directorio personal. Al ser Linux un sistema operativo multiusuario debe ser capaz de ofrecerle a cada usuario un lugar en la organización de

archivos del sistema. En este directorio personal se le estará permitido crear sus propios archivos y directorios. Los programas que él use guardarán acá sus preferencias personales y, en definitiva, toda su actividad personal deberá quedar guardada en su directorio personal.

Como medida de seguridad elemental se le estará permitido el acceso en modo lectura a la mayoría de los archivos y directorios del resto del sistema de archivos. También se le estará permitido el acceso para ejecución de los programas que estén instalados, siempre y cuando no se trate de programas de gestión administrativa del sistema.

Si colocamos el comando **pwd** nos responderá `/home/ubuntu`. Si queremos saber qué contiene utilizamos el comando **ls** para listar a los archivos y directorios, veremos los di- rectorios `Descargas, Desktop, Documentos, Imágenes, Música, Plantillas, Público y Videos`, como vemos en la Figura 9

Figura 9: Primeros comandos

Sabemos que son directorios (carpetas) porque tienen color celeste. Estamos viendo en texto y en línea de comandos lo que también podríamos ver en forma gráfica.

Debajo del botón de Inicio, en el Lanzador de Aplicaciones, está el botón de Archivos, que vemos en la Figura 10:

Figura 10: Botón Archivos

El que, al presionarlo, la aplicación gráfica Nautilus nos mostrará en una ventana (Figura 11) a nuestra Carpeta Personal, cuyo contenido ya conocemos, pero esta vez veremos nuestras subcarpetas de forma gráfica:

Y acá empezamos a ver los puntos en común entre las CLI y las GUI. Si en la ventana

«Terminal» tecleamos el comando nautilus veremos que se lanza la aplicación y nos muestra esta ventana.

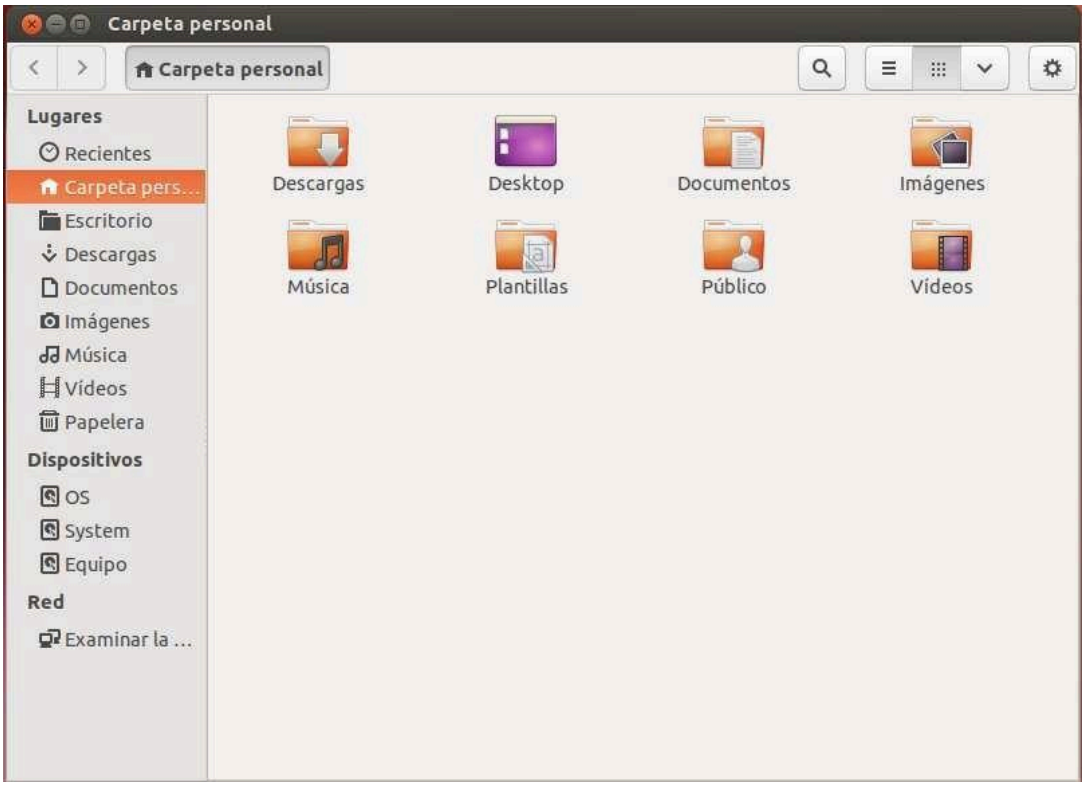

Figura 11: Carpeta personal en Nautilus

El comando dmesg

Una de las referencias más antiguas del comando dmesg (*diagnostic message*) aparece en las páginas del manual de la versión 4.3 BSD (*Berkeley Software Distribution*) Muestra (y controla) el buffer de anillo del núcleo (*kernel ring buffer*). Se le llama
«de anillo» porque es una estructura de datos de tamaño limitado que, al llenarse, se continúa llenando por el principio. Desde el momento del arranque (*boot*) el núcleo guarda ahí todos sus mensajes, sobretodo del hardware que va detectando, de manera que es una buena herramienta para identificar los componentes de nuestra computadora, y luego cada vez que conectamos o desconectamos un dispositivo (por ejemplo al USB) el núcleo registra lo acontecido en ese *buffer*. Lo mismo ocurre si el hardware comienza a fallar - típicamente, nuestro disco rígido-. Como la información que nos muestra por pantalla es muy extensa y se pierde, vamos a verla «por partes». Para ello, vamos a valernos del comando less, usándolos conjuntamente mediante la barra vertical «|» llamada «pleca» en castellano, como vemos en la Figura 12.

```
ubuntu@ubuntu:~$ dmesg | less
```

Con las flechas hacia arriba y hacia abajo nos desplazamos por el archivo, hasta que toquemos la

tecla «q» («*quit*»).

Figura 12: Comando dmesg

procfs

En muchos sistemas operativos tipo UNIX se utiliza el sistema de archivos *proc* (*proc filesystem o procfs*), que es un sistema de archivos, generado dinámicamente, que muestra información de procesos. procfs es un sistema de archivos que no tiene relación con un dispositivo de almacenamiento por bloques. Está implementado en el núcleo del sistema operativo y permite llevar a «espacio de usuario»[7] datos del núcleo y de los procesos, que, sin la ayuda de procfs, sería casi imposible accederlos. Este sistema de archivos está montado normalmente en el directorio /proc y, dado que no contiene archivos reales, no consume espacio de almacenamiento. Este sistema tiene su origen en Killian (1984).

Entonces, podemos ver información acerca de nuestra CPU colocando el comando

```
ubuntu@ubuntu:~$ cat  /proc/cpuinfo
```

Dentro de nuestra ventana «Terminal». cat es un programa que se utiliza para con**cat**enar archivos, pero utilizado de este modo muestra el contenido de un archivo (de texto) por la pantalla.

Si tiene un procesador con doble núcleo (*hyperthreading* o posterior), seguramente verá información acerca de dos procesadores, nombre, frecuencia en MHz, tamaño de la caché y el valor de BogoMIPS (*Bogus Millions Instructions Per Second*), que es una medida no científica de medir la velocidad del procesador y que es típica del núcleo Linux utilizada para calibrar el *busy-loop* o espera activa. Es decir que podríamos calificarlo como «el número de millones de veces por segundo en el que un procesador no puede hacer absolutamente nada».

Para ver información acerca de nuestra memoria, el comando

```
ubuntu@ubuntu:~$ cat  /proc/meminfo
```

Nos mostrará el estado actual de la memoria del sistema, incluyendo información acerca de la

memoria virtual y la caché. Es el principal método que utilizan los programas de aplicación (que operan con el procesador en modo usuario) para obtener información acerca del estado de la memoria. Mayormente, la utilizan los utilitarios que muestran el uso de los recursos del sistema.

Podemos ver información acerca de las IRQs con el comando

```
ubuntu@ubuntu:~$ cat /proc/interrupts
```

que nos muestra el número de la interrupción usada y el nombre de quien la usa.

Los archivos en /proc, a excepción de unos pocos, tienen tamaño cero; esto es así porque en realidad el archivo no contiene datos, sino que actúa como un puntero hacia donde residen los datos. Es «una ventana al núcleo».

Llamadas a sistema

Podemos utilizar el programa strace (*system call trace*) que sigue la traza a las llamadas a sistema y las señales de un proceso, interceptándolas, registrándolas y mostrándolas por pantalla (*stderr, standard error*).

Cada línea mostrada contiene el nombre de la llamada a sistema, seguido por sus argumentos en paréntesis y sus valores retornados. Por ejemplo, si ejecutáramos el comando

```
ubuntu@ubuntu:~$ cat  /dev/null
```

debería mostrarnos el contenido de un archivo (de contenido nulo en este caso) pero que al seguirle el rastro a las llamadas a sistema que efectúa con el comando

```
ubuntu@ubuntu:~$ strace cat /dev/null
```

nos muestra las sucesivas llamadas al sistema que va efectuando el proceso cat. Podemos destacar entre sus llamadas al sistema la que carga el ejecutable en memoria execve("/bin/cat"... y la que efectivamente abre el archivo para ver su contenido open("/dev/null").

Podemos probar también con un programa simple como pwd, que muestra el directorio de trabajo (es decir, el directorio sobre el cual estamos posicionados actualmente) y que logra su objetivo primordial haciendo una llamada a la función getcwd (*get current working directory*).

También podemos trazar la ejecución de ls, que es el comando utilizado para listar los archivos que existen dentro de un directorio (o carpeta en la terminología de Windows) con el comando

```
ubuntu@ubuntu:~$ strace ls
```

Podemos notar que para listar los archivos y directorios que existen en el directorio ac- tual, abre el archivo «.» que contiene el listado de archivos en este directorio. Esto lo hace con open(".") y luego mediante sucesivas llamadas a getdents, obtiene las entradas del directorio actual y finalmente las muestra por pantalla con write(1, "archivos").

hello, world

El primer programa que se escribió en lenguaje C, en la venerable obra de Kernighan y Ritchie (t988) «*The C programming language*», es el que imprimía en la pantalla (terminal)

```
#include <stdio.h>
int main()
  {
     printf("hello, world\ n") ;
     return 0 ;
  }
```

las palabras «hello, world»; con pocas diferencias, sencillamente este:

La línea «#include <stdio.h>» es una directiva para el preprocesador del compilador. Esta directiva le indica al compilador que debe incluir el contenido del archivo de cabeceras llamado «stdio.h» (*standard input and output*) dentro del programa. Este archivo contiene definiciones de funciones tales como «scanf()» y «printf()». Por lo tanto, si utilizamos en nuestro programa alguna de estas funciones y no solicitamos la inclusión del archivo «stdio.h», el compilador no sabrá qué hacer con esa función no definida y dará error en la compilación.

La ejecución del programa C comienza con la función principal «main()». La función de biblioteca «printf()» envía texto formateado a la pantalla. Y la instrucción «return 0» devuelve un cero -que, por convención, indica que no hubo error-, finalizando efectivamente el programa.

Editores de texto

Desde la línea de comandos se puede invocar al editor de texto «vi» (desarrollado originalmente por Bill Joy en el año 1976, su nombre proviene de «visual») para editar un archivo fuente «hello.c». Es muy probable también que en su distribución Linux ya esté preinstalado el editor de texto «nano», que puede resultarle más amigable, porque toda la ayuda que puede necesitar está en la última línea de la pantalla. Muchas distribuciones Linux también traen preinstalado el programa «vimtutor», es decir, un programa que sirve de tutorial para el editor. Si lo invoca mediante el comando «vimtutor es», se abrirá un archivo de texto en castellano que le va explicando -a medida que avanza en su lectura- cómo utilizar este editor de texto, haciendo las prácticas sobre el mismo archivo.

GNU Compiler Collection

La colección de compiladores GNU es un conjunto de compiladores creados por el proyecto GNU. Originalmente GCC (y el comando «gcc») significaba GNU C Compiler (compilador GNU de C), porque sólo compilaba el lenguaje C, pero posteriormente se extendió para compilar C++, Fortran, Ada y otros. Utilizaremos este comando para compilar el archivo fuente «hello.c» recientemente creado con el editor de nuestra preferencia. Simplemente, mediante el comando:

```
ubuntu@ubuntu:~$ gcc hello.c -o hello
```

Se lo compila; la opción «-o» se utiliza para indicar que el archivo de salida (*output*) a generar debe llamarse como se le indica a continuación, caso contrario, el archivo de salida siempre se llamará «a.out», y se lo ejecuta colocando por delante «./» (punto barra), como se aprecia en la Figura 13.

```
ubuntu@ubuntu:~$ gcc  hello.c  -o  hello
ubuntu@ubuntu:~$ ./hello
hello, world
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$
```

Figura 13: hello, world

Podemos trazar las llamadas a sistema que hace este sencillo programa, como sabemos, anteponiendo el comando «strace» al ejecutable, como vemos en la Figura 14.

```
ubuntu@ubuntu:~$ strace ./hello
execve("./hello", ["./hello"], 0xbfcd1ca0 /* 67 vars */) = 0
brk(NULL)                               = 0x1371000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7f08000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=131330, ...}) = 0
mmap2(NULL, 131330, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7ee7000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/i386-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\3\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\20\220\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1942840, ...}) = 0
mmap2(NULL, 1948188, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7d0b000
mprotect(0xb7ee0000, 4096, PROT_NONE)   = 0
mmap2(0xb7ee1000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d5000) = 0xb7ee1000
mmap2(0xb7ee4000, 10780, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb7ee4000
close(3)                                = 0
set_thread_area({entry_number=-1, base_addr=0xb7f090c0, limit=0x0fffff, seg_32bit=1, contents=0, read_exec_only=0,
er=6)
mprotect(0xb7ee1000, 8192, PROT_READ)   = 0
mprotect(0x438000, 4096, PROT_READ)     = 0
mprotect(0xb7f35000, 4096, PROT_READ)   = 0
munmap(0xb7ee7000, 131330)              = 0
fstat64(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(136, 1), ...}) = 0
brk(NULL)                               = 0x1371000
brk(0x1392000)                          = 0x1392000
brk(0x1393000)                          = 0x1393000
write(1, "hello, world\n", 13hello, world
)          = 13
exit_group(0)                           = ?
+++ exited with 0 +++
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$
```

Figura 14.: strace de hello, world

También podemos generar el código assembler de nuestro programa C indicando la opción «-S», de esta manera:

```
ubuntu@ubuntu:~$ gcc -S hello.c
```

Este comando invoca al preprocesador (cpp) sobre el archivo hello.c, realiza una compilación inicial y luego se detiene antes de ejecutar el ensamblador. Genera el archivo hello.s.

Disponer del código assembler nos ayudará a comprender mejor cómo se llevará a cabo este conjunto de instrucciones (programa) en el procesador y la memoria.

En Python

El programa equivalente escrito en lenguaje Python sería simplemente:

```
#!/usr/bin/env python
print("hello, world")
```

Si lo edita en el archivo «hello.py», por ejemplo, puede luego ejecutarlo desde la línea de comandos.

El comando time

Puede tomarle el tiempo de ejecución a los comandos anteponiéndoles el comando «time». De la misma manera que hicimos con «strace»:

```
ubuntu@ubuntu:~$ time  ./hello
hello, world
real 0m0.107s user  0m0.003s sys 0m0.000s
ubuntu@ubuntu:~$
```

Haga estas prácticas tanto con el ejecutable del código C como con el de Python. **Observe** las diferencias. **Compare** las dos ejecuciones.

Práctica con Windows

Para muchas prácticas con Windows vamos a valernos de un grupo de herramientas desarrolladas por Mark Russinovich, quien fundó una empresa llamada Sysinternals y cuyo sitio en Internet es

https://learn.microsoft.com/en-us/sysinternals/

Es cierto que Windows provee de ciertas herramientas para ver información acerca del procesador y de la memoria. Por ejemplo, en Windows XP la información acerca del procesador puede verla en el «Administrador de dispositivos», al cual se accede fácilmente desde la pestaña de «Hardware» en la «Propiedades del sistema», ícono «Mi PC» y «Ver información del sistema».

Pero con los utilitarios de Sysinternals puede ver información mucho más específica para la cual no siempre necesita privilegios de administrador, es suficiente con los privilegios de la cuenta Invitado. Varios de estos utilitarios deben ejecutarse desde una ventana de «Símbolo de sistema», es decir que los utilitarios no se basan en una GUI sino en una CLI.

También contamos con las Herramientas de Soporte de Windows o *Windows Support Tools*, que son unas 40 herramientas útiles para administrar y descubrir problemas en los sistemas Windows. Se las puede instalar ejecutando el programa setup.exe en la carpeta \Support\Tools en el CD de distribución de Windows.

Por otra parte, contamos con los Equipos de Recursos de Windows (*Windows Resource Kits*) que suplementan a las Herramientas de Soporte de Windows porque agregan herramientas adicionales para la administración del sistema. Se pueden descargar gratuitamente de

http://www.microsoft.com/

buscando *"resource kit tools*".

Adicionalmente, necesitaremos para el mejor desempeño de ciertos utilitarios de Sysinternals, a las Herramientas de Depuración de Windows o *Windows Debugging Tools*, que podemos descargar gratuitamente de

https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-tools

Estas herramientas normalmente quedan instaladas en

\Archivos de programa\Debugging Tools for Windows (x86)\

Además, en algunos casos, podemos compilar sencillos programas que se ofrecen tanto para Linux como para Windows. Como ya sabemos, la distribución Ubuntu que hemos elegido trae el compilador libre gcc. Para Windows también tenemos compiladores libres (o al menos gratuitos) del lenguaje C. Por ejemplo, el compilador Borland en Windows. He aquí las instrucciones para

dejarlo operativo:

https://developerinsider.co/download-and-install-borland-c-compiler-on-windows-10/

ProcFeatures

Por ejemplo, el utilitario ProcFeatures de Sysinternals (basado en CLI) utiliza la API

IsProcessorFeaturePresent

de Windows para determinar si el procesador y Windows soportan distintos rasgos, tales como páginas de no-ejecución (*No-Execute pages*), extensiones de direcciones físicas (*Physical Address Extensions o PAE*) y un contador de ciclo de tiempo real. Su propósito principal es identificar la versión de PAE del núcleo que ejecuta el sistema y si soporta la protección de desborde del *buffer* de no-ejecución. Además, nos muestra el tipo de procesador (marca y modelo), la frecuencia en MHz, si soporta el conjunto de instrucciones MMX y SSE -o *Streaming SIMD Extensions*- de Intel.

Ejemplo

```
C:\Archivos de programa\Sysinternals>ProcFeatures
Process Feature v1.10
Copyright (C) 2005 Mark Russinovich
Sysinternals - www.sysinternals.com
                    Intel(R) Pentium(R) 4 CPU 3.00GHz
x86 Family 15 Model 4 Stepping 1, GenuineIntel No
Execute Protection:                         N
Physical Address Extensions (PAE): N
Floating point emulation:

N Pentium Floating point errata:

N RDTSC (Cycle counter):

Y
MMX Instruction Set:                        Y
3D Now Instruction Set:                     N
SSE Instruction Set:                        Y
SSE2 Instruction Set:

Y C:\Archivos de programa\Sysinternals>
```

Ejemplo

```
C:\Archivos de programa\Sysinternals>ProcFeatures Process Feature v1.10
  Copyright (C) 2005 Mark Russinovich
  Sysinternals - www.sysinternals.com
  AMD Athlon(tm) 64 X2 Dual Core Processor 5200+ x86
  Family 15 Model 107 Stepping 2, AuthenticAMD No
  Execute Protection:                         Y
  Physical Address Extensions (PAE): Y
```

```
Floating point emulation:

N Pentium Floating point errata:

N RDTSC (Cycle counter):

Y
MMX Instruction Set:                    Y
3D Now Instruction Set:                 Y
SSE Instruction Set:                    Y
SSE2 Instruction Set:

Y C:\Archivos de programa\SysInternals>
```

CoreInfo

CoreInfo es un utilitario de línea de comando que nos muestra la equivalencia entre procesadores lógicos y procesadores físicos, nodo NUMA y el *socket* en el cual residen, como así también la *caché* asignada a cada procesador. Utiliza la función de Windows

GetLogicalProcessorInformation

para obtener esta información y mostrarla por pantalla. CoreInfo es útil para obtener una visión dentro del procesador y la topología de *caché* de su sistema.

Ejemplo

```
C:\Archivos de programa\Sysinternals>coreinfo
Coreinfo v1.00 - Dump information on CPU cores, NUMA nodes, sockets and caches
Copyright (C) 2008 Mark Russinovich
Sysinternals - www.sysinternals.com
Logical to Physical Processor Map:
**    Physical Processor 0 (Hyperthreaded)
Logical Processor to Socket Map:
Logical Processor to NUMA Node Map:
** NUMA Node 0
Logical Processor to Cache Map:
C:\Archivos de programa\Sysinternals>
```

Ejemplo

```
C:\Archivos de programa\Sysinternals>Coreinfo
Coreinfo v1.00 - Dump information on CPU cores, NUMA nodes, sockets and caches Copyright
(C) 2008 Mark Russinovich
Sysinternals - www.sysinternals.com
Logical to Physical Processor Map:
*-    Physical Processor 0
```

```
-*    Physical Processor 1
Logical Processor to Socket Map:
**    Socket 0
Logical Processor to NUMA Node Map:
**    NUMA Node 0
Logical Processor to Cache Map:
*-    Data Cache          0, Level 1,   64 KB, Assoc  2, LineSize   64
*-    Instruction Cache   0, Level 1,   64 KB, Assoc  2, LineSize   64
*-    Unified Cache       0, Level 2,  512 KB, Assoc 16, LineSize   64
-*    Data Cache          1, Level 1,   64 KB, Assoc  2, LineSize   64
-*    Instruction Cache   1, Level 1,   64 KB, Assoc  2, LineSize   64
-*    Unified Cache       1, Level 2,  512 KB, Assoc 16, LineSize   64
C:\Archivos de programa\Sysinternals>
```

Llamadas a sistema

StraceNT es un trazador de llamadas a sistema (*System Call Tracer*) para Windows. Provee una funcionalidad similar a la que provee strace para Linux. Puede trazar todas las llamadas hechas por un proceso a las funciones importadas desde otras DLLs. StraceNT puede ser muy útil para depurar y analizar el funcionamiento interno de un programa. Muestra nombres de funciones, sus argumentos y valores devueltos. Para hacer el trazado de llamadas a funciones StraceNT se vale de la sección IAT (*Import Address Table*) del ejecutable PE que veremos en detalle en **??**. Lo podemos obtener de:

http://www.softpedia.com/get/Programming/Debuggers-Decompilers-Dissasemblers/Strace%20NT.shtml

Texe es una herramienta que podemos utilizar para visualizar las Tablas de Importación y Exportación de los ejecutables con formato PE. Lo podemos obtener de:

https://en.freedownloadmanager.org/Windows-PC/Texe-FREE.html

Administrador de tareas

El Administrador de tareas o *Task Manager* de Windows nos permite ver la actividad de las aplicaciones en modo usuario y el efecto en modo núcleo. En la pestaña «Rendimiento» y luego seleccionando del menú «Ver» la opción «Mostrar cronología del núcleo». La barra de uso de la CPU mostrará el uso total de la CPU en un color (por ejemplo verde) y el tiempo en modo núcleo en otro (por ejemplo rojo) para atender las llamadas a sistema.

El compilador Dev-C++

Para compilar nuestros códigos en lenguaje C vamos a utilizar este compilador, que es un IDE (*Integrated Development Environment*, en inglés, o Entorno de Desarrollo Integrado) libre, portable y rápido. Podemos descargarlo desde:

https://sourceforge.net/projects/orwelldevcpp/