

Práctica de Laboratorio 1

Instalación y Configuración de Máquinas Virtuales

Estudiantes:

Wílmer E. León

Código: 1520010896

Jesús Orlando Orjuela

Código: 100384722

Hugo Alejandro Mejía

Código: 100312289

Fabián Andrés Cabana

Código: 1620010455

Docente:

José León León

Institución Universitaria Politécnico Grancolombiano

Facultad de Ingeniería, Diseño e Innovación

Sistemas Operacionales - Grupo B04 | Grupo de trabajo 11

Bogotá D.C., Colombia

1 de noviembre de 2025

Índice

1. Introducción	2
2. Marco teórico	2
3. Contenido	2
3.1. Preparación de los recursos	2
3.2. Creación de las máquinas virtuales	3
3.3. Configuración de red	3
3.4. Evidencia de configuración de red	4
3.5. Evidencia de conectividad entre máquinas	6
3.6. Diagramas de red	8
4. Implementación con Docker	8
5. Implementación con Docker	9
6. Conclusiones	10
7. Referencias	10
Referencias	10

1. Introducción

En esta primera entrega se documenta la instalación de tres máquinas virtuales utilizando **Oracle VirtualBox**, su configuración de red en modo **adaptador puente** y la verificación de conectividad entre ellas. El objetivo es sentar las bases para la posterior configuración de **Nginx** como balanceador de carga.

Como señalan autores como Stallings (2005), la virtualización es un pilar fundamental en la administración moderna de sistemas. Asimismo, Silberschatz, Galvin, y Gagne (2008) destacan que la concurrencia y la gestión de procesos son elementos centrales en el diseño de sistemas operativos.

2. Marco teórico

- La virtualización permite optimizar recursos y aislar entornos de ejecución, lo que facilita la administración de sistemas (Wolf, Ruiz, Bergero, y Meza, 2010). Para este caso en particular, el montaje de tres sistemas operativos desde distintas máquinas, se efectúa para hacer pruebas de conectividad.
- Existen diferencias clave entre NAT y adaptador puente en VirtualBox, que determinan el alcance de la conectividad de las máquinas virtuales (Oracle, 2024).
- El direccionamiento IP es esencial para garantizar la comunicación entre dispositivos en una red local (Canonical, 2024).

3. Contenido

3.1. Preparación de los recursos

Se utilizaron imágenes ISO de **Ubuntu Server**, en particular la versión **22.04.5 LTS** para la máquina **UbunSO2**. El hipervisor empleado fue **Oracle VirtualBox** (Oracle, 2024).

3.2. Creación de las máquinas virtuales

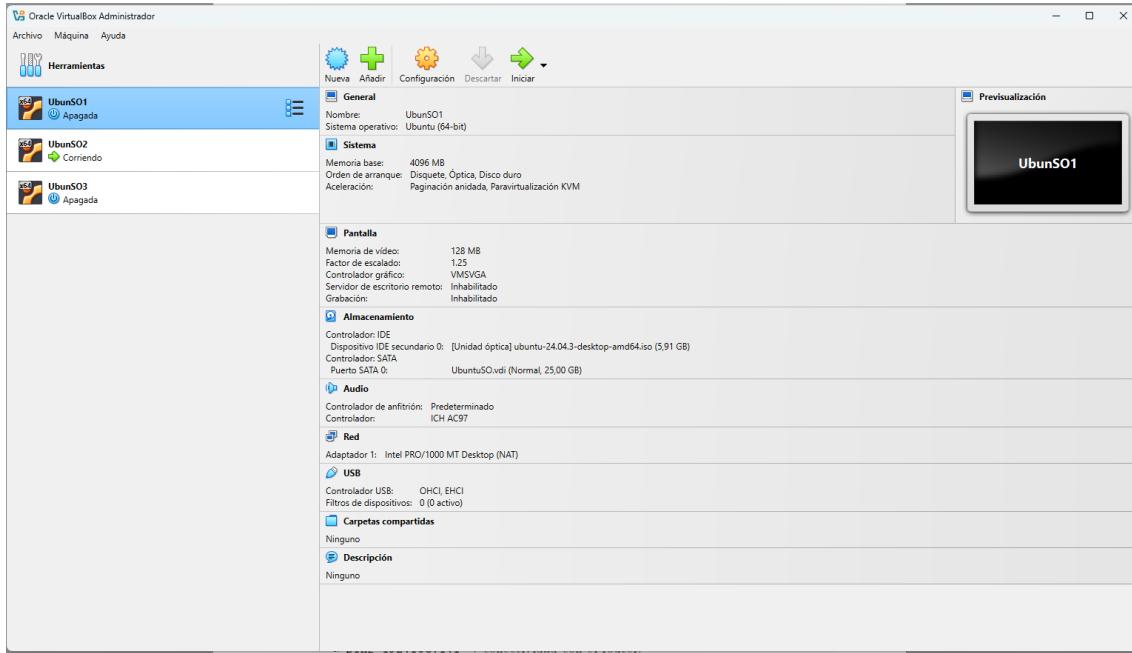


Figura 3.1. Evidencia de las tres máquinas virtuales creadas en VirtualBox.

3.3. Configuración de red

Tabla 3.1. Resumen de configuración de las máquinas virtuales.

Máquina	Nombre	S. O.	RAM (MB)	Dirección IP
UbunSO1	Ubuntu Server 22.04 LTS	Ubuntu Server 22.04 LTS	2048	192.168.2.9
UbunSO2	Ubuntu Server 22.04.5 LTS	Ubuntu Server 22.04.5 LTS	2048	192.168.2.8
UbunSO3	Ubuntu Server 22.04 LTS	Ubuntu Server 22.04 LTS	2048	192.168.2.7

3.4. Evidencia de configuración de red

Para verificar la configuración de red de cada máquina virtual, se utilizó el comando `ip a` (abreviatura de `ip address`) en cada una de las tres máquinas. Este comando muestra información detallada sobre las interfaces de red, incluyendo las direcciones IP asignadas, máscaras de subred y estado de las interfaces.

Pasos realizados:

1. Acceder a cada máquina virtual a través de la consola de VirtualBox.
2. Ejecutar el comando: `ip a` o `ip address show`
3. Verificar que la interfaz de red (típicamente `enp0s3`) tenga asignada la dirección IP configurada.
4. Confirmar que el estado de la interfaz sea UP y que esté en modo `BROADCAST, MULTICAST`.

A continuación se muestran las capturas de pantalla con la salida del comando en cada máquina virtual:

```

ubuntu@ubuntu-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:62:ba:ed brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.9/24 brd 192.168.2.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe62:baed/64 scope link dadfailed tentative
        valid_lft forever preferred_lft forever
ubuntu@ubuntu-VirtualBox:~$ 
```

(a) *UbunSO1 – IP 192.168.2.9*

```

ubuntu@ubuntu-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:62:ba:ed brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.8/24 brd 192.168.2.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe62:baed/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu-VirtualBox:~$ 
```

(b) *UbunSO2 – IP 192.168.2.8*

```

ubuntu@ubuntu-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:62:ba:ed brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.7/24 brd 192.168.2.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe62:baed/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu-VirtualBox:~$ 
```

(c) *UbunSO3 – IP 192.168.2.7*

Figura 3.2. Salida del comando *ip a* en las tres máquinas virtuales.

3.5. Evidencia de conectividad entre máquinas

Una vez verificada la configuración de red, se procedió a comprobar la conectividad entre las máquinas virtuales utilizando el comando `ping`. Este comando envía paquetes ICMP (Internet Control Message Protocol) para verificar si existe comunicación entre dos dispositivos en la red.

Procedimiento para las pruebas de conectividad:

1. Desde UbunSO1 hacia UbunSO2:

- Acceder a la máquina UbunSO1
- Ejecutar: `ping 192.168.2.8 -c 4`
- El parámetro `-c 4` indica que se enviarán 4 paquetes
- Verificar que se reciban respuestas (reply) y no haya pérdida de paquetes

2. Desde UbunSO2 hacia UbunSO3:

- Acceder a la máquina UbunSO2
- Ejecutar: `ping 192.168.2.7 -c 4`
- Confirmar respuestas exitosas y tiempo de respuesta (TTL)

3. Desde UbunSO3 hacia UbunSO1:

- Acceder a la máquina UbunSO3
- Ejecutar: `ping 192.168.2.9 -c 4`
- Validar conectividad bidireccional entre todas las máquinas

Las siguientes capturas muestran los resultados de las pruebas de conectividad, donde se observa el tiempo de respuesta (time), el TTL (Time To Live) y la estadística final con 0 % de pérdida de paquetes:

```

ubuntu@ubuntu-VirtualBox:~$ ping 192.168.2.7
64 bytes from 192.168.2.7: icmp_seq=16 ttl=64 time=1.34 ms
^C
--- 192.168.2.7 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 2108
8ms
rtt min/avg/max/mdev = 0.716/3.474/8.745/2.641 ms
ubuntu@ubuntu-VirtualBox:~$ 

ubuntu@ubuntu-VirtualBox:~$ ping 192.168.2.8
^C
--- 192.168.2.8 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 2171
9ms
rtt min/avg/max/mdev = 1.629/3.688/12.142/2.609 ms
ubuntu@ubuntu-VirtualBox:~$ ^C
ubuntu@ubuntu-VirtualBox:~$ 

```

(a) $UbuSO1 \rightarrow UbuSO2$

```

ubuntu@ubuntu-VirtualBox:~$ ping 192.168.2.9
64 bytes from 192.168.2.9: icmp_seq=24 ttl=64 time=1.82 ms
64 bytes from 192.168.2.9: icmp_seq=25 ttl=64 time=2.92 ms
^C
--- 192.168.2.9 ping statistics ---
28 packets transmitted, 25 received, 10.7143% packet loss, time
27114ms
rtt min/avg/max/mdev = 1.204/2.366/9.931/1.621 ms
ubuntu@ubuntu-VirtualBox:~$ 

ubuntu@ubuntu-VirtualBox:~$ ping 192.168.2.7
64 bytes from 192.168.2.7: icmp_seq=48 ttl=64 time=1.36 ms
64 bytes from 192.168.2.7: icmp_seq=49 ttl=64 time=2.65 ms
64 bytes from 192.168.2.7: icmp_seq=50 ttl=64 time=1.78 ms
64 bytes from 192.168.2.7: icmp_seq=51 ttl=64 time=1.08 ms
^C
--- 192.168.2.7 ping statistics ---
51 packets transmitted, 49 received, 3.92157% packet loss, time
50143ms
rtt min/avg/max/mdev = 0.709/1.737/3.832/0.741 ms
ubuntu@ubuntu-VirtualBox:~$ 

```

(b) $UbuSO2 \rightarrow UbuSO3$

```

ubuntu@ubuntu-VirtualBox:~$ ping 192.168.2.9
^C
--- 192.168.2.9 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 1914
2ms
rtt min/avg/max/mdev = 0.565/1.702/3.180/0.754 ms
^C
ubuntu@ubuntu-VirtualBox:~$ 

ubuntu@ubuntu-VirtualBox:~$ ping 192.168.2.8
^C
--- 192.168.2.8 ping statistics ---
24 packets transmitted, 24 received, 0% packet loss, time 23670
ms
rtt min/avg/max/mdev = 0.757/1.512/2.764/0.614 ms
ubuntu@ubuntu-VirtualBox:~$ 

```

(c) $UbuSO3 \rightarrow UbuSO1$

Figura 3.3. Pruebas de conectividad entre las máquinas virtuales mediante `ping`.

3.6. Diagramas de red

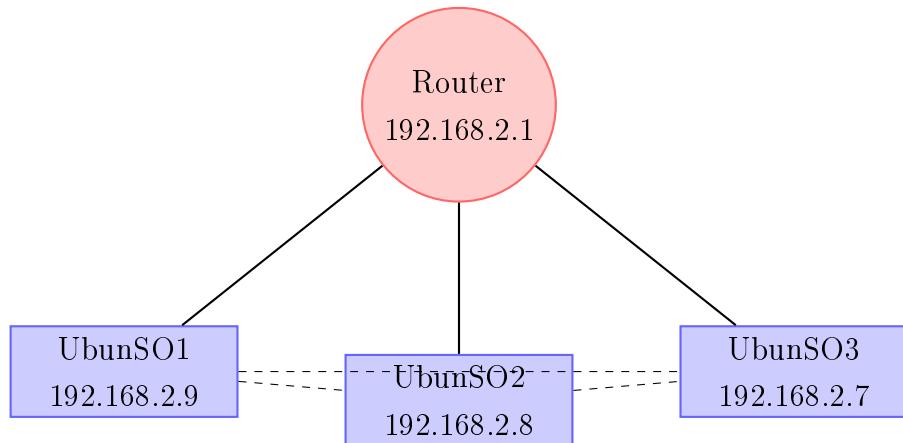


Figura 3.4. Comunicación entre las máquinas y el router en modo Adaptador Puente.

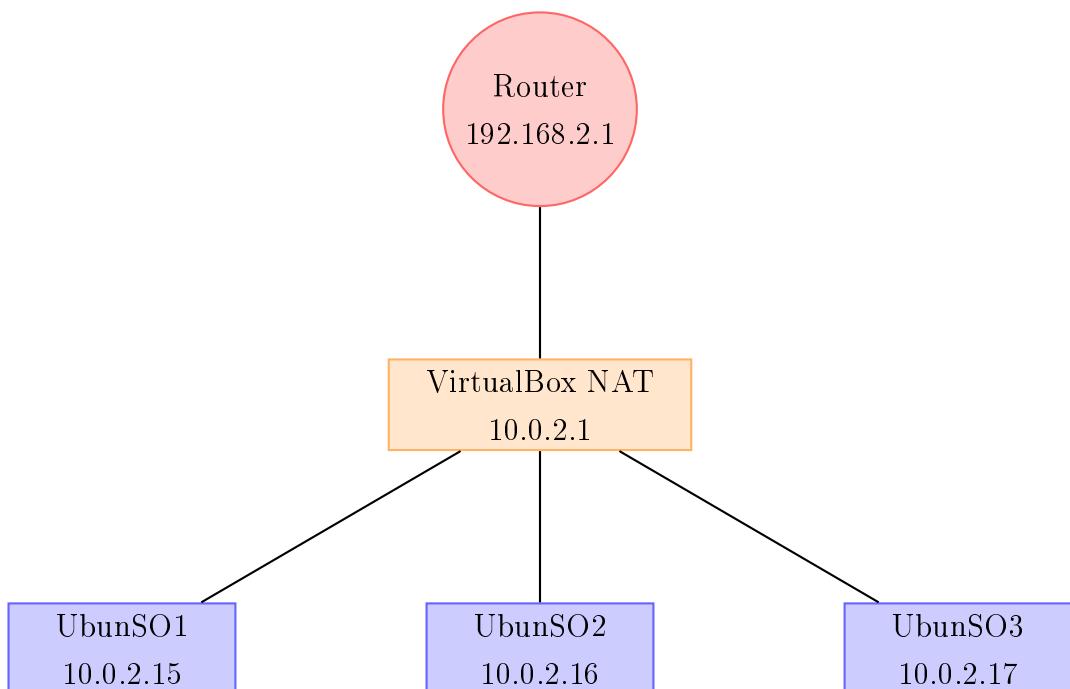


Figura 3.5. Esquema de red en modo NAT (referencia).

4. Implementación con Docker

Generamos además una implementación aprovechando la potencia de Docker para generar virtualización a partir de sus contenedores y teniendo en cuenta la importancia que tiene esta herramienta en el mundo profesional. Se generan contenedores y red via bash y podemos verlos ejecutando Ubuntu 22.04:

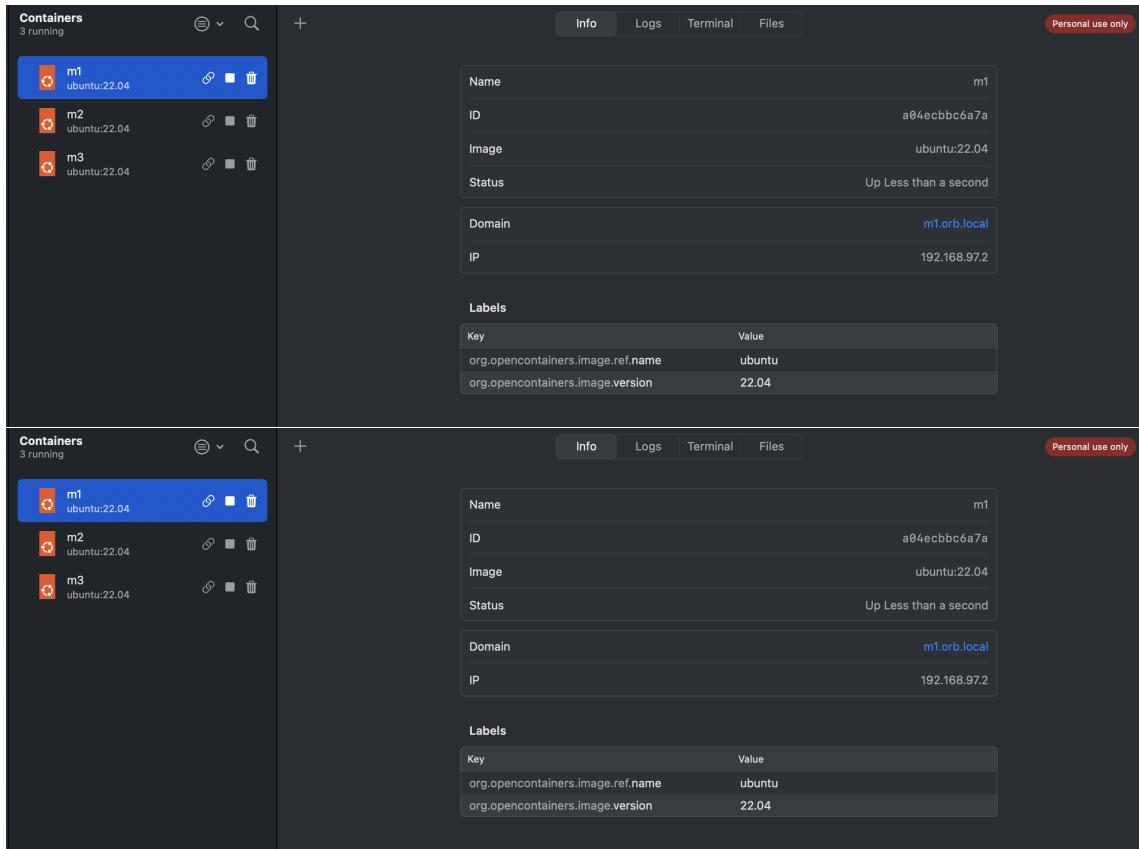


Figura 4.1. Captura del software OrbStack

5. Implementación con Docker

Generamos además una implementación aprovechando la potencia de Docker para generar virtualización a partir de sus contenedores y teniendo en cuenta la importancia que tiene esta herramienta en el mundo profesional. Se generan contenedores y red via bash y podemos verlos ejecutando Ubuntu 22.04:

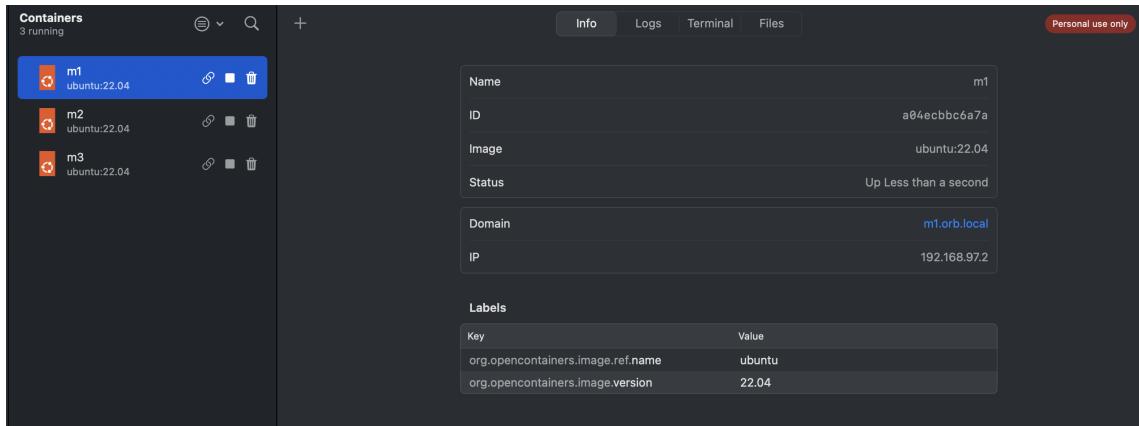


Figura 5.1. Captura del software OrbStack

6. Conclusiones

La configuración en modo puente permitió que cada VM tuviera una IP propia en la red local, facilitando la comunicación entre ellas y con el router. Las pruebas de ping confirmaron la conectividad entre las máquinas, sin pérdida de paquetes significativa. La asignación de IPs fijas mediante Netplan garantiza estabilidad para futuras pruebas y la siguiente fase con Nginx como balanceador de carga.

De acuerdo con Canonical (2024), la documentación oficial de Ubuntu recomienda este tipo de configuraciones para entornos de laboratorio. Asimismo, Oracle (2024) enfatiza la importancia de comprender las diferencias entre NAT y adaptador puente en VirtualBox para escenarios de red. Por su parte, Arena (2002) y Arena (2005) destacan la relevancia de Linux como sistema operativo robusto para servidores y entornos académicos, mientras que Wolf y cols. (2010) subraya su papel en la enseñanza universitaria.

7. Referencias

Referencias

- Arena, H. F. (2002). *La biblia de linux*. Buenos Aires: MP Ediciones.
- Arena, H. F. (2005). *La biblia de linux: Manual de uso, instalación y configuración*. Buenos Aires: RedUsers.
- Canonical. (2024). Ubuntu server documentation [Manual de software informático]. Descargado de <https://ubuntu.com/server/docs>
- Oracle. (2024). Virtualbox user manual [Manual de software informático]. Descargado de <https://www.virtualbox.org/manual/>
- Silberschatz, A., Galvin, P. B., y Gagne, G. (2008). *Fundamentos de sistemas operativos* (7a ed. ed.). México: McGraw-Hill.

Stallings, W. (2005). *Sistemas operativos: Aspectos internos y principios de diseño* (5a ed. ed.). Madrid: Pearson Prentice Hall.

Wolf, G., Ruiz, E., Bergero, F., y Meza, E. (2010). *Fundamentos de sistemas operativos*. México: Universidad Nacional Autónoma de México.