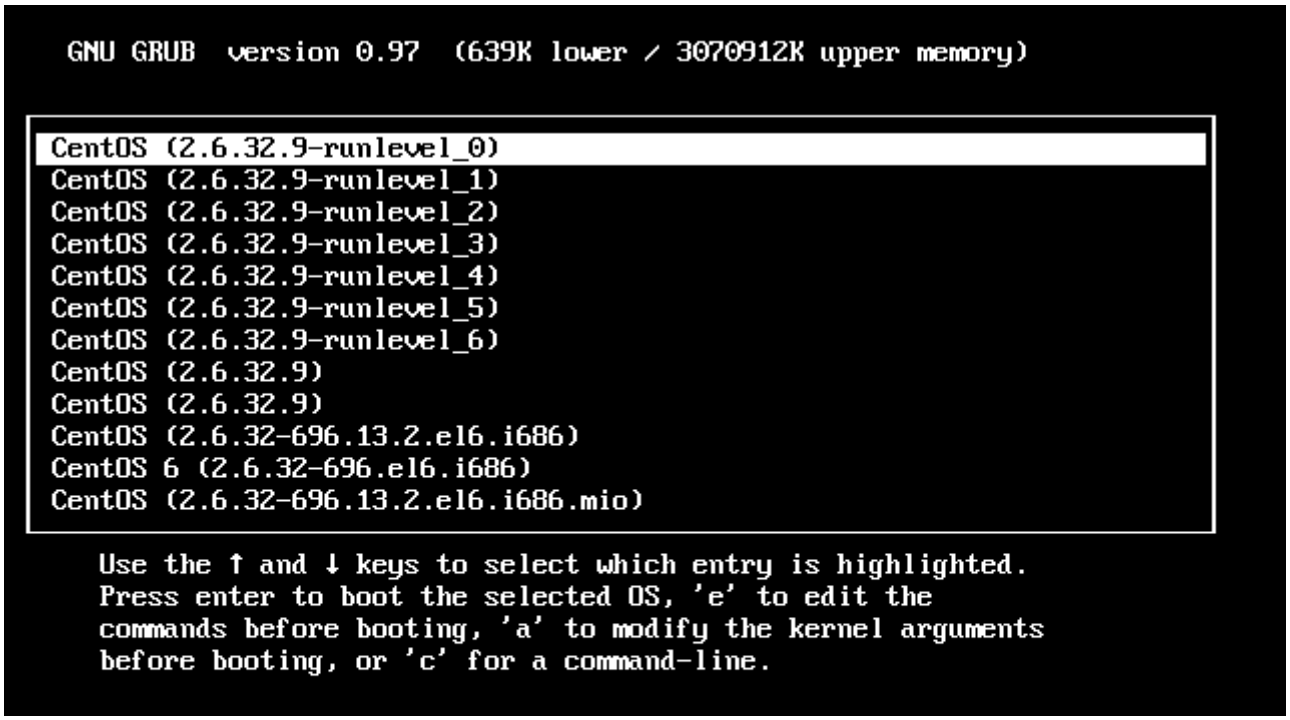
Memoria Práctica 3

Administración de Servidores

Gestión del arranque de un sistema GNU/Linux: Init y GRUB



Almudena García Jurado-Centurión

Índice

0. Introducción

El arranque del sistema es el proceso que permite iniciar nuestro sistema operativo, preparándolo para ser utilizado por los usuarios.

En los sistemas GNU/Linux, este proceso de arranque es realizado por un programa llamado *Init*, que va iniciando los diferentes procesos necesarios para poner en marcha el sistema.

En algunos *init* como *sysvinit,* estos procesos están organizados según diferentes niveles de arranque o *runlevels*, scripts que definen los programas a ejecutar durante el arranque.
Estos niveles de arranque pueden ser configurados por el administrador del sistema, modificando los niveles ya existentes o creando otros nuevos.

Pero, antes de lanzar el Init, necesitamos otro programa que indique a la máquina la ubicación del sistema operativo que queremos iniciar, e inicie el proceso para cargar su kernel en memoria. Este programa es conocido como *bootloader* o Gestor de Arranque.

Al igual que sucede con el *Init,* el gestor de arranque puede ser configurado por el usuario, añadiendo entradas para otros sistemas operativos, o para nuevos kernel de sistemas GNU/Linux ya existentes.
En estas entradas también se pueden indicar opciones de arranque, como parámetros del kernel o el nivel de arranque en que se quiere iniciar el sistema.

En ésta práctica aprenderemos a personalizar estos dos programas, añadiendo nuevos procesos a los niveles de arranque, y creando nuevas entradas en el bootloader que inicien el sistema en varios de estos niveles.

1. Creando un nuevo servicio en el init

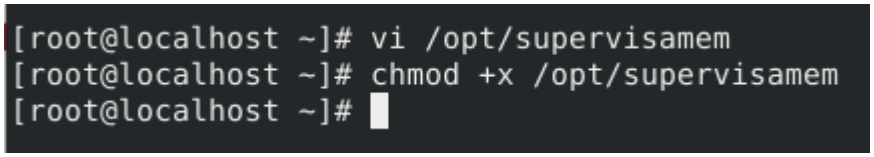
En este apartado, crearemos un nuevo servicio en el init llamado *supervisamem*. Este servicio ejecutará un script del mismo nombre, alojado en */opt/supervisamem*

El código del script será el siguiente:

```
#!/bin/bash
while true

do

 ahora = $(date "+%H:%M:%S - ")

 echo -n $ahora >> /var/log/supervisamem.log

 grep Dirty /proc/meminfo >> /var/log/supervisamem.log

 sleep 30

done
```

Este script se irá ejecutando cada 30 segundos, obteniendo información sobre el estado de la memoria, junto a la hora a la que se tomó la medición, y guardándola en un fichero llamado *supervisamem.log,* ubicado en */var/log*

Copiamos el código en un nuevo fichero en */opt/supervisamem.* Vemos que hay un pequeño error en la cuarta linea (en las asignaciones a variables no se admiten espacios), así que aprovechamos para corregirlo. Guardamos el fichero ya corregido, y le damos permisos de ejecución.



Una vez creado el script a ejecutar, debemos crear un nuevo servicio que lo ejecute.
Para ello, debemos crear un script de inicio, que tendrá las instrucciones a ejecutar en cada orden que reciba el servicio.

Nos basaremos en el ejemplo de este tutorial, cambiando los valores indicados por los correspondientes a nuestro script:

https://www.cyberciti.biz/tips/linux-write-sys-v-init-script-to-start-stop-service.html

El script de inicio estará alojado en */etc/init.d,* con el mismo nombre que el script, y su contenido será similar a este:

```
#!/bin/bash
#
# chkconfig: 35 90 12
# description: memory monitor
#

# Get function from functions library
. /etc/init.d/functions

# Start the service FOO
start() {
        initlog -c "echo -n Starting supervisamem server: "
        /opt/supervisamem &
        ### Create the lock file ###
        touch /var/lock/subsys/supervisamem
        success $"supervisamem server startup"
        echo
}

# Restart the service supervisamem
stop() {
        initlog -c "echo -n Stopping supervisamem server: "
        killproc supervisamem
        ### Now, delete the lock file ###
        rm -f /var/lock/subsys/supervisamem
        echo
}

### main logic ###
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
```

```
        ### Create the lock file ###
        touch /var/lock/subsys/supervisamem
        success $"supervisamem server startup"
        echo
}

# Restart the service supervisamem
stop() {
        initlog -c "echo -n Stopping supervisamem server: "
        killproc supervisamem
        ### Now, delete the lock file ###
        rm -f /var/lock/subsys/supervisamem
        echo
}

### main logic ###
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  status)
        status supervisamem
        ;;
  restart|reload|condrestart)
        stop
        start
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart|reload|status}"
        exit 1
esac

exit 0
```

Una vez tenemos el script de inicio, podemos iniciar el servicio escribiendo:

/etc/init.d/supervisamem start

Y, para detenerlo:

/etc/init.d/supervisamem stop

También podemos obtener el estado del servicio con:

/etc/init.d/supervisamem status

Probamos a iniciar nuestro servicio, para comprobar que funcione correctamente

```
[root@localhost ~]# /etc/init.d/supervisamem start
/etc/init.d/supervisamem: línea 12: initlog: no se encontró la orden
                                                           [  OK  ]
[root@localhost ~]# ps -e | grep supervisa
 1918 pts/0    00:00:00 supervisamem
[root@localhost ~]# ls /var/log | grep supervisamem
supervisamem.log
[root@localhost ~]# more /var/log/supervisamem.log
16:36:10 -Dirty:                 0 kB
16:36:40 -Dirty:                40 kB
[root@localhost ~]#
```

Vemos que, aunque se ha mostrado un pequeño error, el servicio se ha iniciado correctamente y ha generado el log.

Ahora probamos a detenerlo:

```
[root@localhost ~]# /etc/init.d/supervisamem stop
/etc/init.d/supervisamem: línea 22: initlog: no se encontró la orden
                                                           [  OK  ]
[root@localhost ~]# ps -e | grep supervisa
[root@localhost ~]#
```

Vemos que el servicio se ha detenido correctamente, y que su proceso asociado ya no existe.

```
[root@localhost ~]# /etc/init.d/supervisamem status
supervisamem está parado
[root@localhost ~]# /etc/init.d/supervisamem start
/etc/init.d/supervisamem: línea 12: initlog: no se encontró la orden
                                                           [  OK  ]
[root@localhost ~]# /etc/init.d/supervisamem status
Se está ejecutando supervisamem (pid 1977)...
[root@localhost ~]# /etc/init.d/supervisamem stop
/etc/init.d/supervisamem: línea 22: initlog: no se encontró la orden
                                                           [  OK  ]
[root@localhost ~]# /etc/init.d/supervisamem status
supervisamem está parado
[root@localhost ~]#
```

Vemos que la orden *status* también funciona correctamente.

2. Gestionando servicios con *chkconfig*

En este paso, vamos a configurar los niveles de arranque en los que se iniciará el servicio, y comprobar los enlaces simbolicos creados para tal fin.

Para esto, usaremos la herramienta *chkconfig,* que nos permite gestionar los servicios del sistema desde linea de comandos.

Previamente, para que el servicio pueda ser configurado con *chkconfig,* tenemos que añadirlo con *chkconfig - -add [nombre_servicio]*

```
[root@localhost ~]# chkconfig --add supervisamem
[root@localhost ~]#
```

Una vez añadido el servicio a *chkconfig, e*mpezaremos añadiendo el servicio a los niveles 2, 4 y 5; mediante la orden:

chkconfig - - level [niveles] [nombre_servicio] on/off

Donde *niveles* se corresponderá a los niveles de arranque donde los queremos añadir o quitar; *nombre_servicio* se corresponderá al nombre del servicio que queremos, y *on/off* indica si lo queremos añadir o quitar de dichos niveles.

En nuestro caso, la orden quedará así:

```
[root@localhost ~]# chkconfig --level 245 supervisamem on
[root@localhost ~]#
```

Tras ejecutar la orden, comprobamos los enlaces simbólicos que se han establecido en los directorios de cada nivel de arranque; en este caso, el 2, 4 y 5

Para cada nivel de arranque hay un directorio, ubicado en */etc/rc.d/rcX.d,* siendo X el número correspondiente a cada nivel de arranque.
Dentro de cada uno de esos directorios, están los enlaces simbólicos a los scripts de inicio de los servicios que tienen asociados.

En este caso, si todo ha ido bien, deberemos encontrar un enlace simbólico llamado *supervisamem* en los directorios *rc2.d*, *rc4.d* y *rc5.d*

```
[root@localhost ~]# ls -l /etc/rc.d/rc2.d/ | grep supervisamem
lrwxrwxrwx. 1 root root 22 nov  6 16:50 S90supervisamem -> ../init.d/supervisa
mem
[root@localhost ~]#
[root@localhost ~]# ls -l /etc/rc.d/rc4.d/ | grep supervisamem
lrwxrwxrwx. 1 root root 22 nov  6 16:50 S90supervisamem -> ../init.d/supervisamem
[root@localhost ~]# ls -l /etc/rc.d/rc5.d/ | grep supervisamem
lrwxrwxrwx. 1 root root 22 nov  6 16:50 S90supervisamem -> ../init.d/supervisamem
[root@localhost ~]#
```

Vemos que los enlaces simbólicos se han establecido correctamente, apuntando a */etc/init.d/supervisamem*, donde está ubicado el script de inicio del servicio.

2.1. Comprobando la configuración de nuestro servicio

Ahora, vamos a comprobar el nivel por defecto en que se ejecuta la máquina, y si nuestro servicio esta incluido en él.

Para ver el nivel de arranque por defecto, abrimos el fichero */etc/inittab* y buscamos la palabra *initdefault.*

```
[root@localhost ~]# grep initdefault /etc/inittab
#   0 - halt (Do NOT set initdefault to this)
#   6 - reboot (Do NOT set initdefault to this)
id:3:initdefault:
```

En este caso, vemos que el nivel de arranque por defecto es el 3. Nuestro servicio está en los niveles 2, 4 y 5, así que no pertenece al nivel de arranque por defecto.

Una vez comprobado que nuestro servicio no está en el nivel por defecto, nos cambiamos al nivel 2 para comprobar que nuestro servicio se inicia en ese nivel.

Para cambiar el nivel de arranque, usaremos el comando *telinit [nivel]*, siendo *nivel* el nivel de arranque al que queremos cambiar.

En nuestro caso, usaremos *telinit 2.*

Una vez ejecutado el comando *telinit,* ejecutaremos el comando *runlevel*, que nos muestra el nivel anterior y nuestro nivel actual.

```
[root@localhost ~]# telinit 2
[root@localhost ~]# runlevel
3 2
[root@localhost ~]#
```

En nuestro caso, vemos que nuestro anterior nivel era el 3, y que actualmente estamos en el 2.

Tras cambiar al nivel 2, comprobamos que el servicio *supervisamem* se ha iniciado correctamente.

```
[root@localhost ~]# ps -e | grep supervisamem
 1374 ?        00:00:00 supervisamem
[root@localhost ~]#  /etc/init.d/supervisamem status
Se está ejecutando supervisamem (pid 1374)...
[root@localhost ~]#
```

2.2. Eliminando nuestro servicio de los niveles de arranque

Hecho esto, volvemos al nivel por defecto (el nivel 3), y eliminamos el servicio *supervisamem* de todos los niveles de arranque.

```
[root@localhost ~]# telinit 3
[root@localhost ~]# runlevel
2 3
[root@localhost ~]#
```

Para elliminar el servicio en todos los niveles de arranque, usamos:

chkconfig [nombre_servicio] off

```
[root@localhost ~]# chkconfig supervisamem off
[root@localhost ~]# init 4
[root@localhost ~]# ps -e | grep supervisamem
[root@localhost ~]# init 2
[root@localhost ~]# ps -e | grep supervisamem
[root@localhost ~]# init 5
[root@localhost ~]# ps -e | grep supervisamem
[root@localhost ~]#
```

Tras eliminar el servicio, vemos que ya no está en ninguno de los niveles que añadimos anteriormente.

Una vez eliminado el servicio, enviamos un mensaje a los usuarios y reiniciamos la máquina. Para ello, usaremos el comando *shutdown,* con sintaxis:

shutdown [opcion] [tiempo] [mensaje]

En nuestro caso, la opción será *-r,* para reiniciar la máquina; el tiempo será *now,* para indicar que se realice ahora; y el mensaje será *“Se ha eliminado el servicio supervisamem”*

El comando quedará así:

```
[root@localhost ~]# shutdown -r now "Se ha eliminado el servicio supervisamem"

Broadcast message from root@localhost.localdomain
        (/dev/pts/0) at 21:15 ...

The system is going down for reboot NOW!
Se ha eliminado el servicio supervisamem
```

Comprobamos que se efectúa el reinicio, mostrando el mensaje indicado.

3. Personalizando el bootloader para diferentes niveles de arranque

En GRUB, es posible editar las entradas del núcleo para indicar el nivel de arranque que queremos iniciar.

En este apartado, crearemos una entrada por cada nivel de arranque disponible. Para ello, usaremos las entradas del menú indicadas en el *grub.conf*

Nuestro *grub.conf* actual es el siguiente:

```
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/VolGroup-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=1
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.32.9)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
        initrd /initramfs-2.6.32.9.img
title CentOS (2.6.32.9)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
        initrd /miRAMDisk.img
title CentOS (2.6.32-696.13.2.el6.i686)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-696.13.2.el6.i686 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrh
eb-sun16 crashkernel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
        initrd /initramfs-2.6.32-696.13.2.el6.i686.img
title CentOS 6 (2.6.32-696.el6.i686)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-696.el6.i686 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-su
n16 crashkernel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
        initrd /initramfs-2.6.32-696.el6.i686.img
title CentOS (2.6.32-696.13.2.el6.i686.mio)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-696.13.2.el6.i686.mio ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latar
cyrheb-sun16 crashkernel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
        initrd /initramfs-2.6.32-696.13.2.el6.i686.img.mio
```

Para especificar el nivel de arranque de un núcleo, debemos poner su número en la línea *kernel* de su entrada.

Si quisiéramos especificar el nivel 3 en la primera entrada, la línea kernel quedará así:

kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashkernel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 3

Los niveles de arranque disponibles son 7:

- 0 : Apagado del sistema
- 1: Monousuario (solo root), sin red ni daemons
- 2: Multiusuario sin soporte de red
- 3: Multiusuario con soporte de red
- 4: Multiusuario con soporte de red (configurable)
- 5: Multiusuario gráfico
- 6: Reinicio del sistema

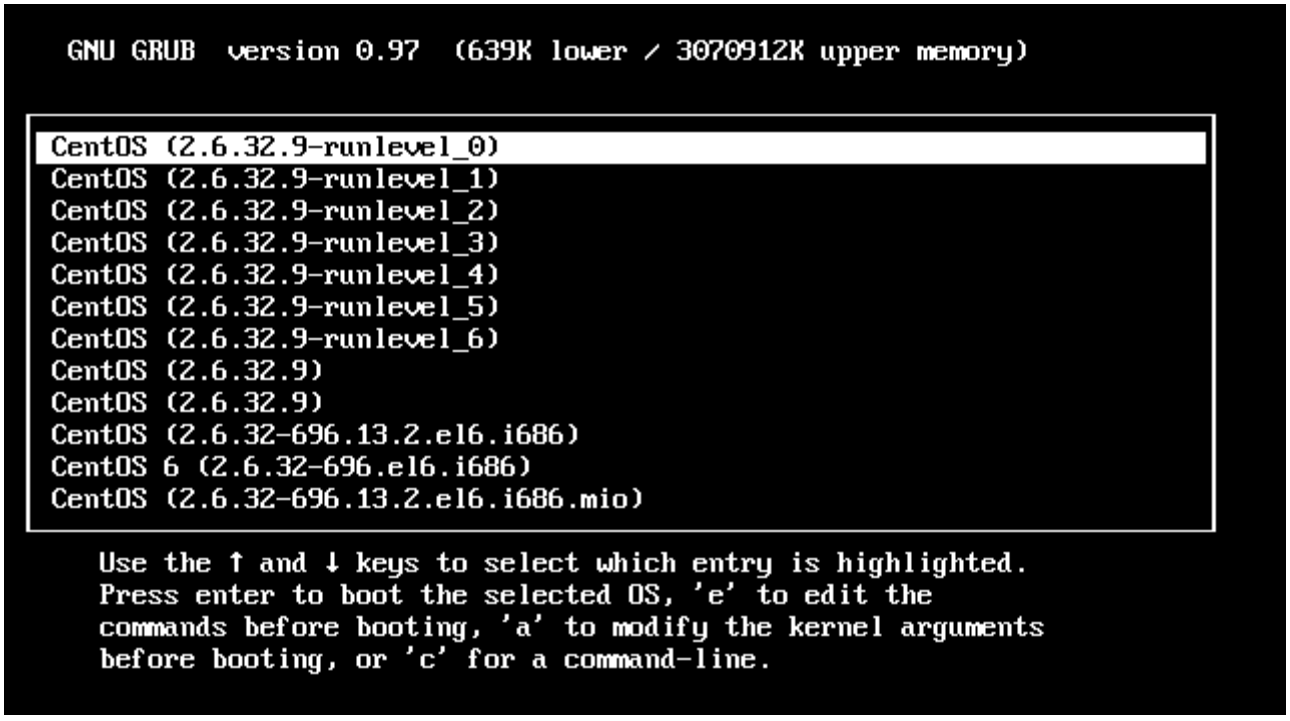
Así pues, tenemos que añadir 7 entradas, una por cada nivel de arranque

Para hacerlo, copiaremos la primera línea 7 veces, cada una con un nivel de arranque diferente.

El fichero resultante será similar a este:

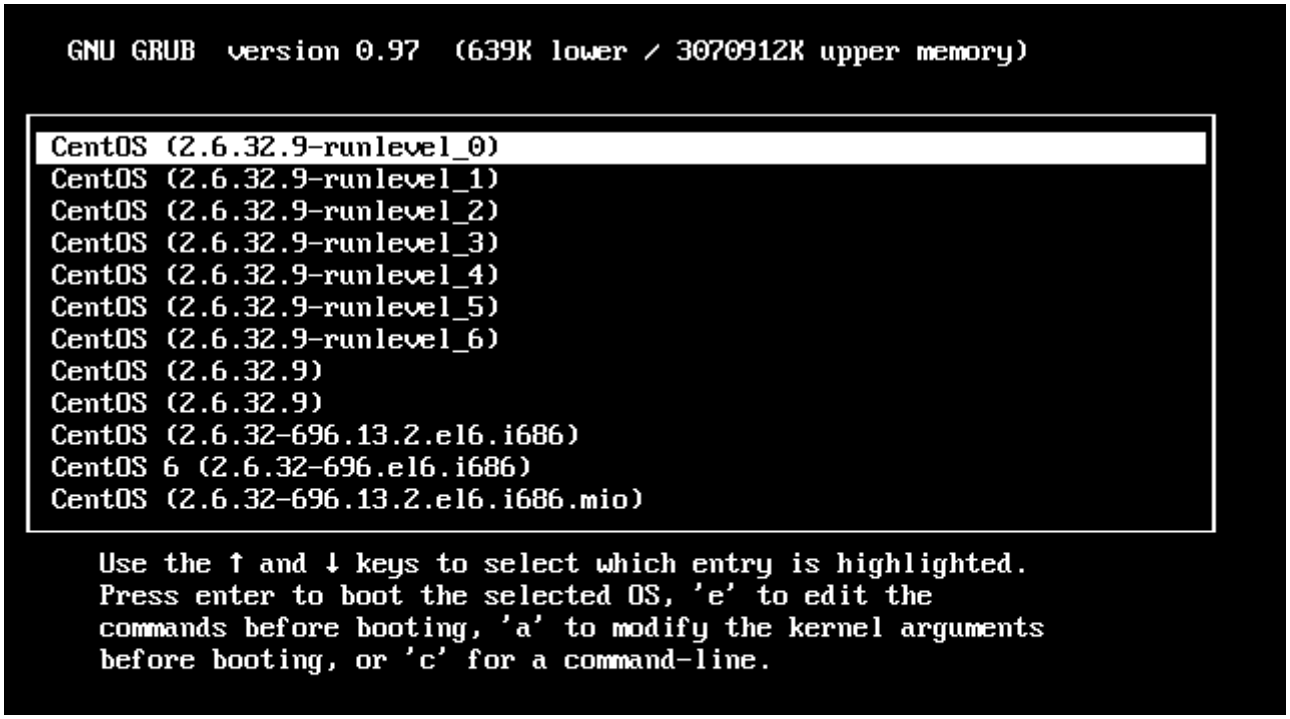
```
title CentOS (2.6.32.9-runlevel_0)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 0
        initrd /initramfs-2.6.32.9.img

title CentOS (2.6.32.9-runlevel_1)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 1
        initrd /initramfs-2.6.32.9.img

title CentOS (2.6.32.9-runlevel_2)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 2
        initrd /initramfs-2.6.32.9.img

title CentOS (2.6.32.9-runlevel_3)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 3
        initrd /initramfs-2.6.32.9.img

title CentOS (2.6.32.9-runlevel_4)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 4
        initrd /initramfs-2.6.32.9.img

title CentOS (2.6.32.9-runlevel_5)
        root (hd0,0)
        kernel /vmlinuz-2.6.32.9 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashke
rnel=auto LANG=es_ES.UTF-8 rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet 5
        initrd /initramfs-2.6.32.9.img

-- INSERT --
```

Tras reiniciar vemos lo siguiente:



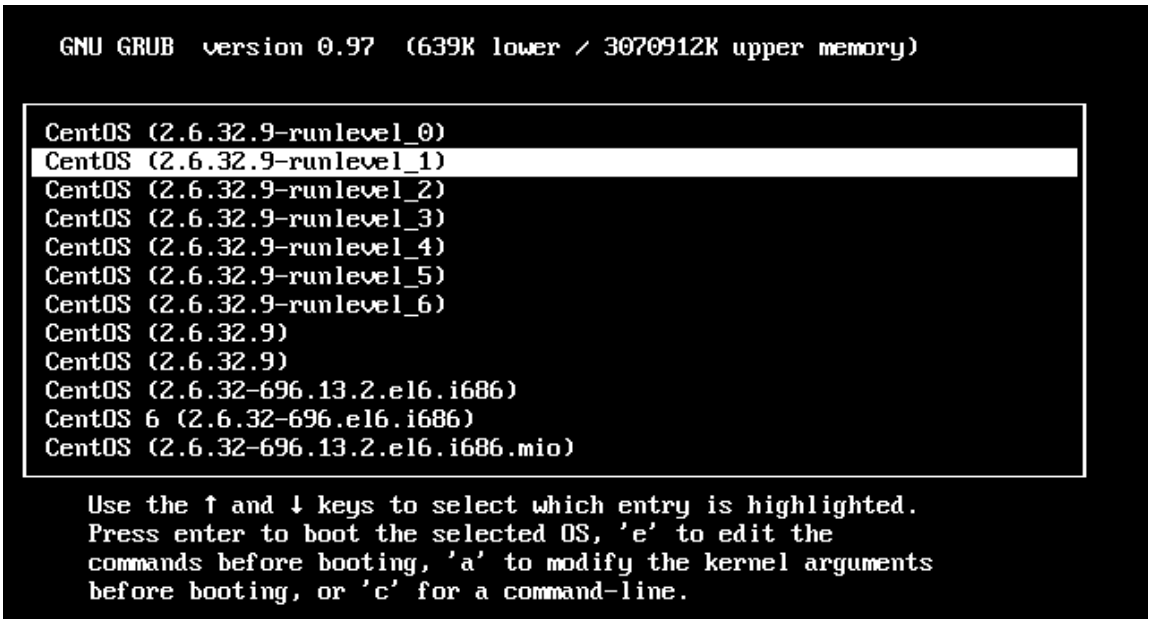
- **Nivel de arranque 0: Apagado**

  Pulsamos en la primera opción, correspondiente al nivel de arranque 0 para ver el resultado. Si todo va bien, debería apagarse el sistema.

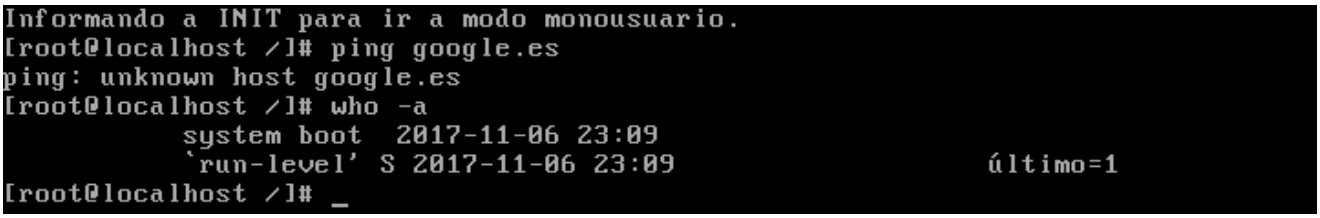


Vemos que el sistema ignora el runlevel y entra al nivel por defecto:

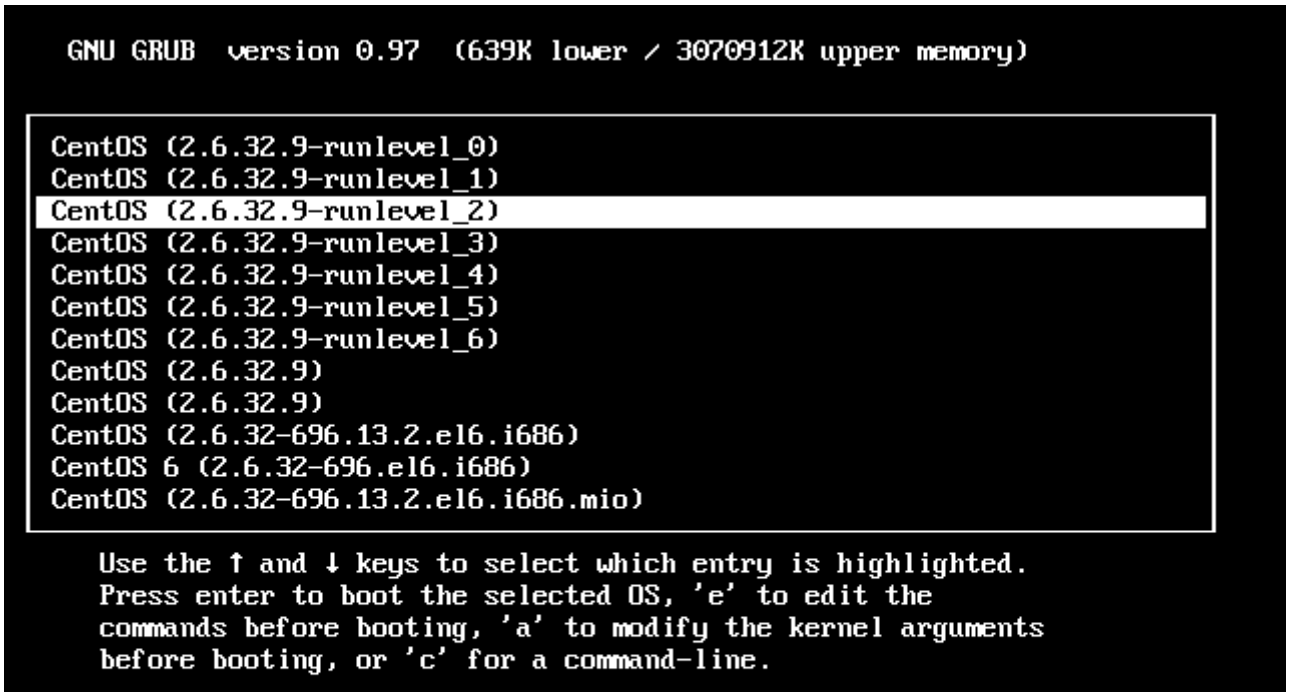
- **Nivel de arranque 1: Monousuario**



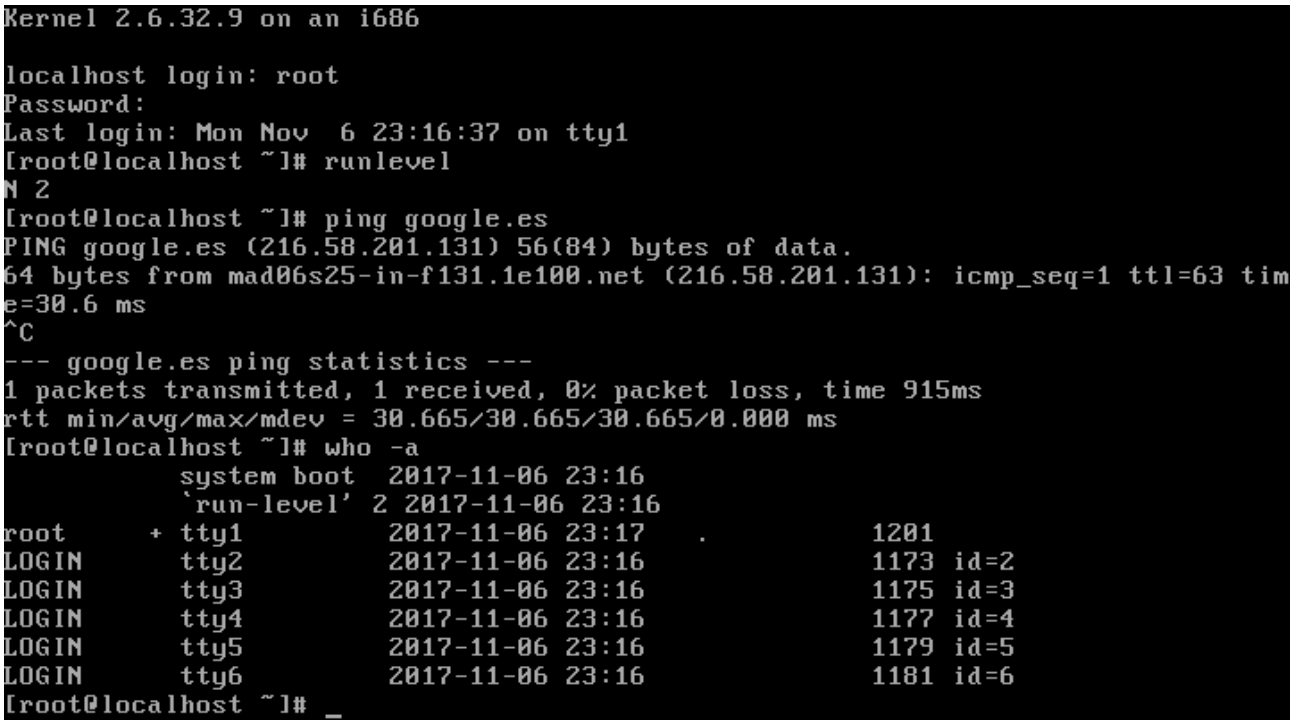
El sistema inicia en modo monousuario, sin soporte de red



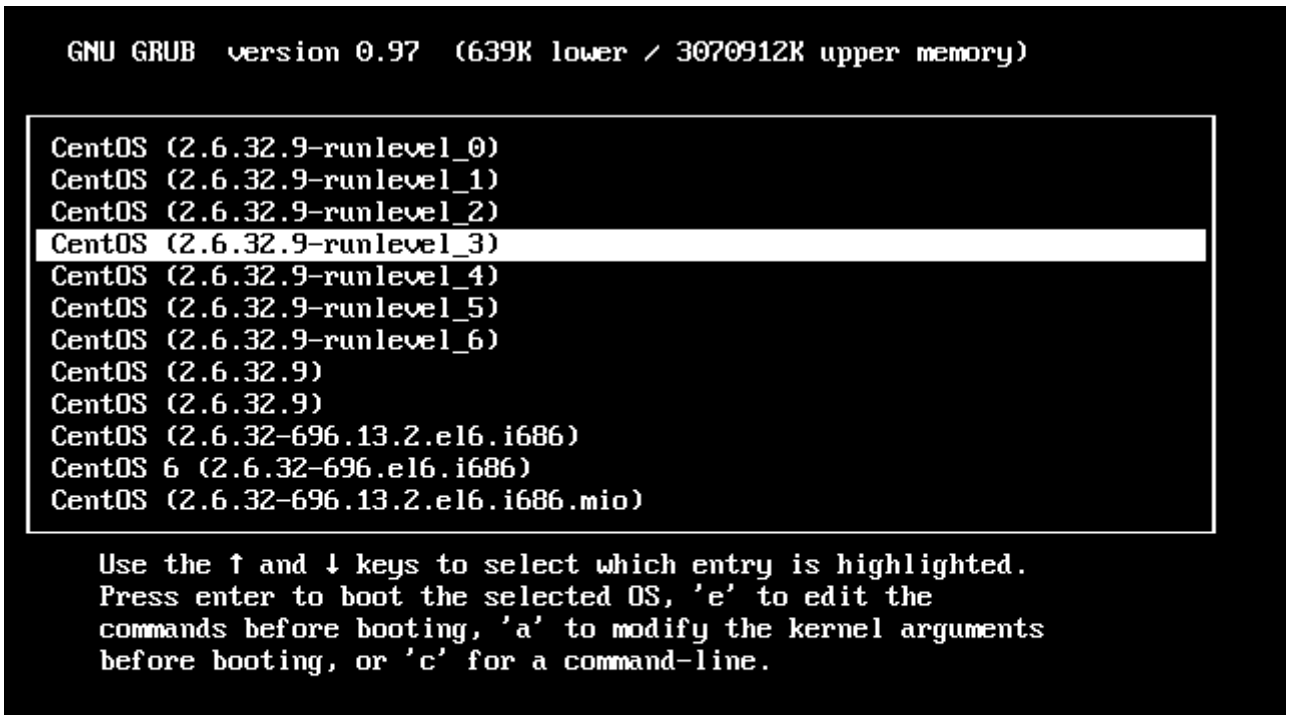
- **Nivel de arranque 2: Multiusuario sin soporte de red**



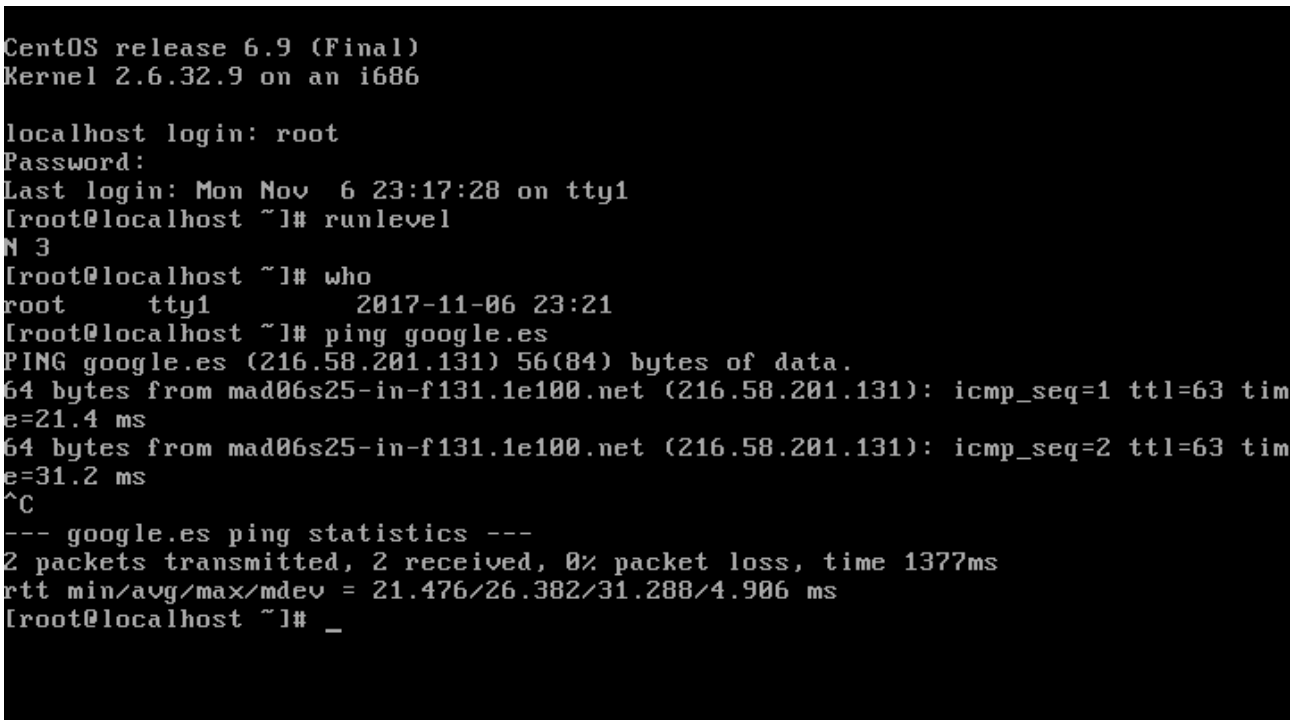
Vemos que, aunque teóricamente no debería tener soporte de red, en éste caso si lo tiene. Comprobamos además que iniciamos en modo multiusuario.



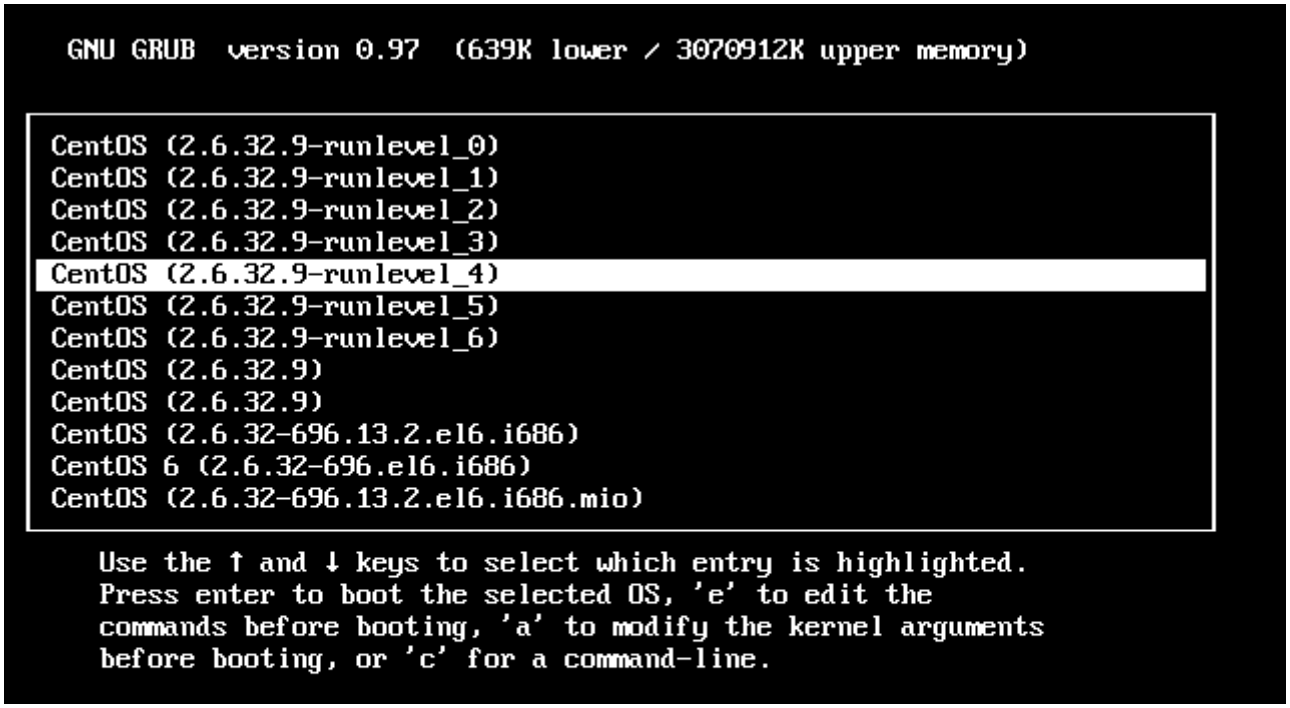
- **Nivel de arranque 3: Multiusuario con soporte de red**



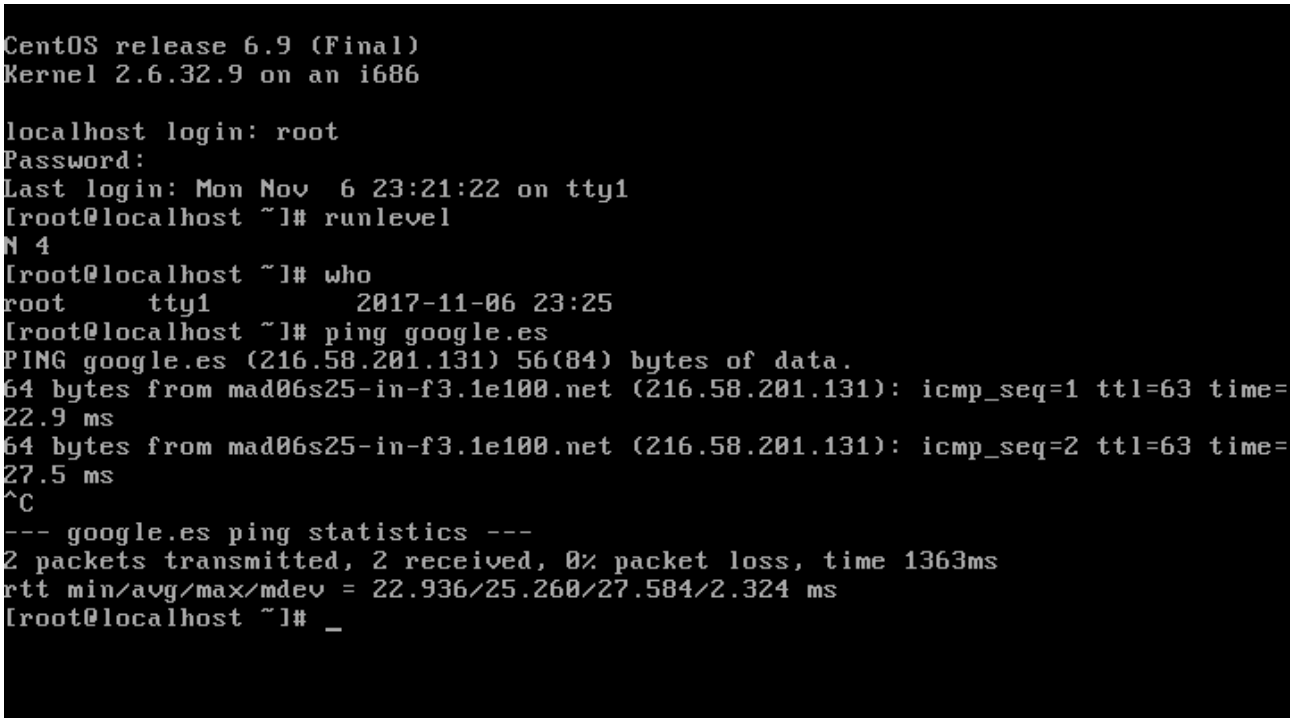
Comprobamos que estamos como multiusuario con soporte de red



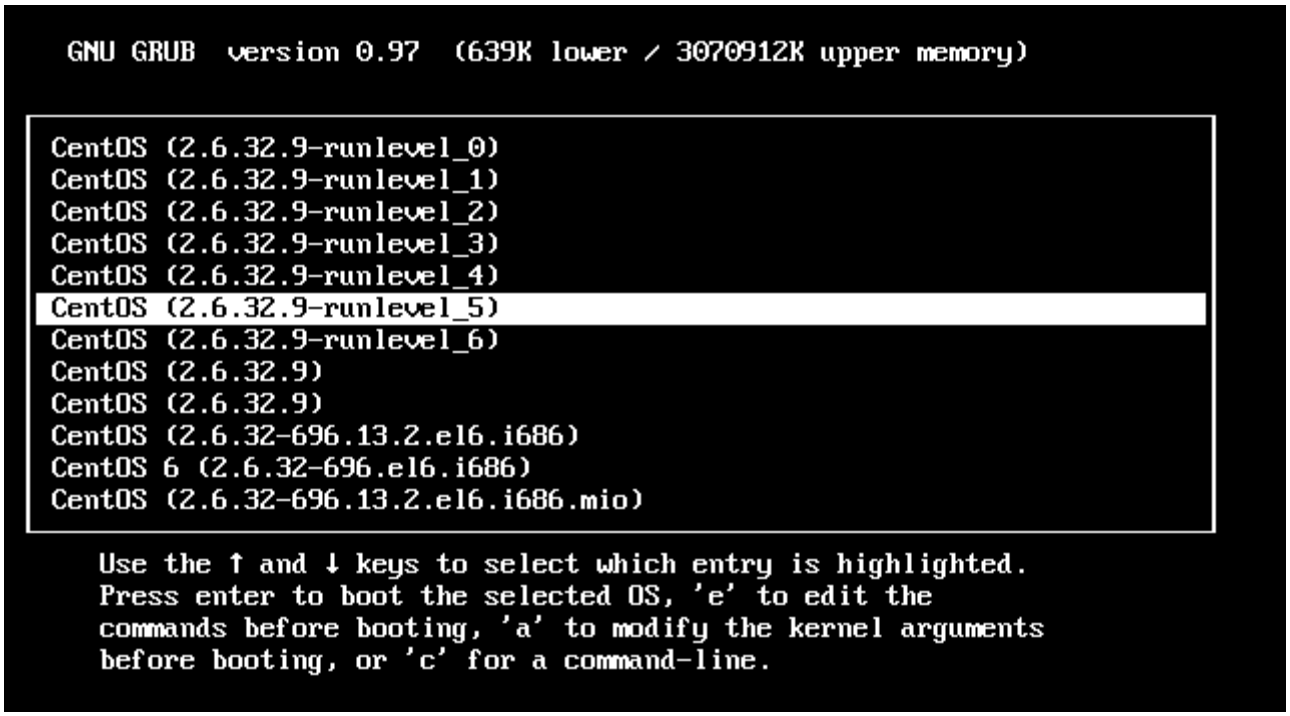
- **Nivel de arranque 4: Multiusuario con soporte de red, personalizable**



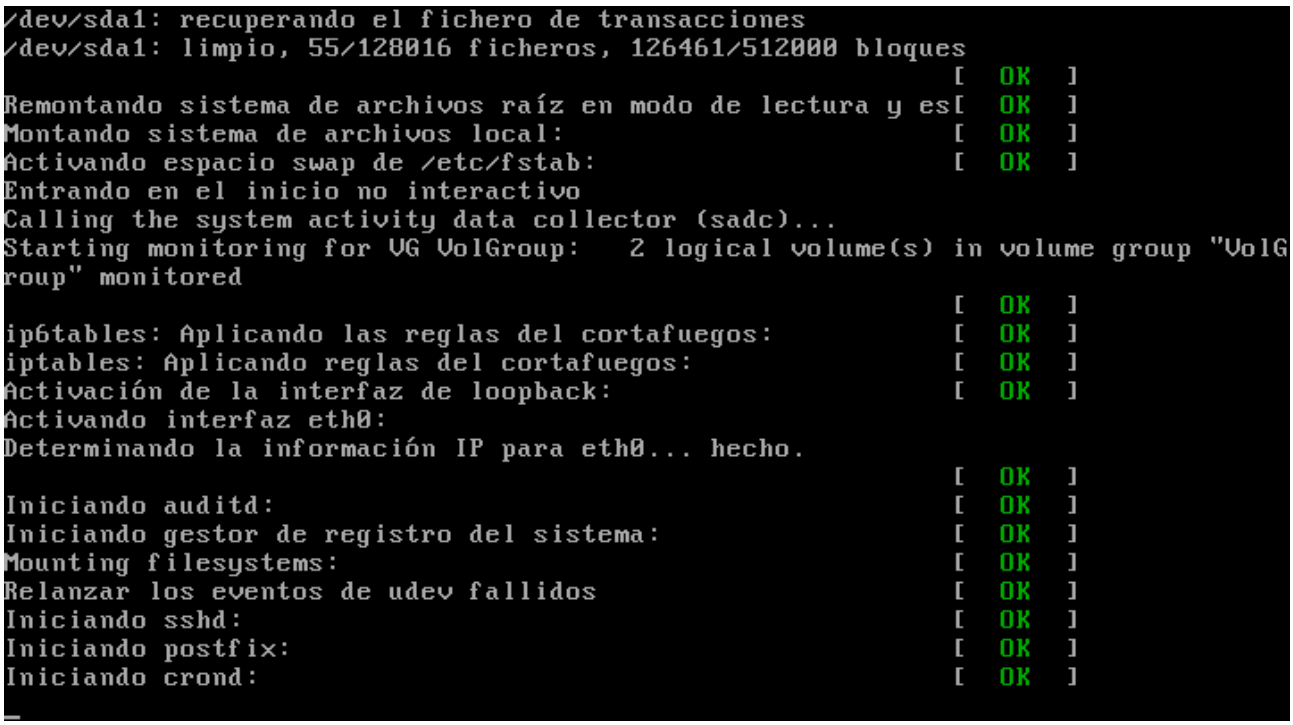
No vemos ninguna diferencia respecto al nivel de arranque 3



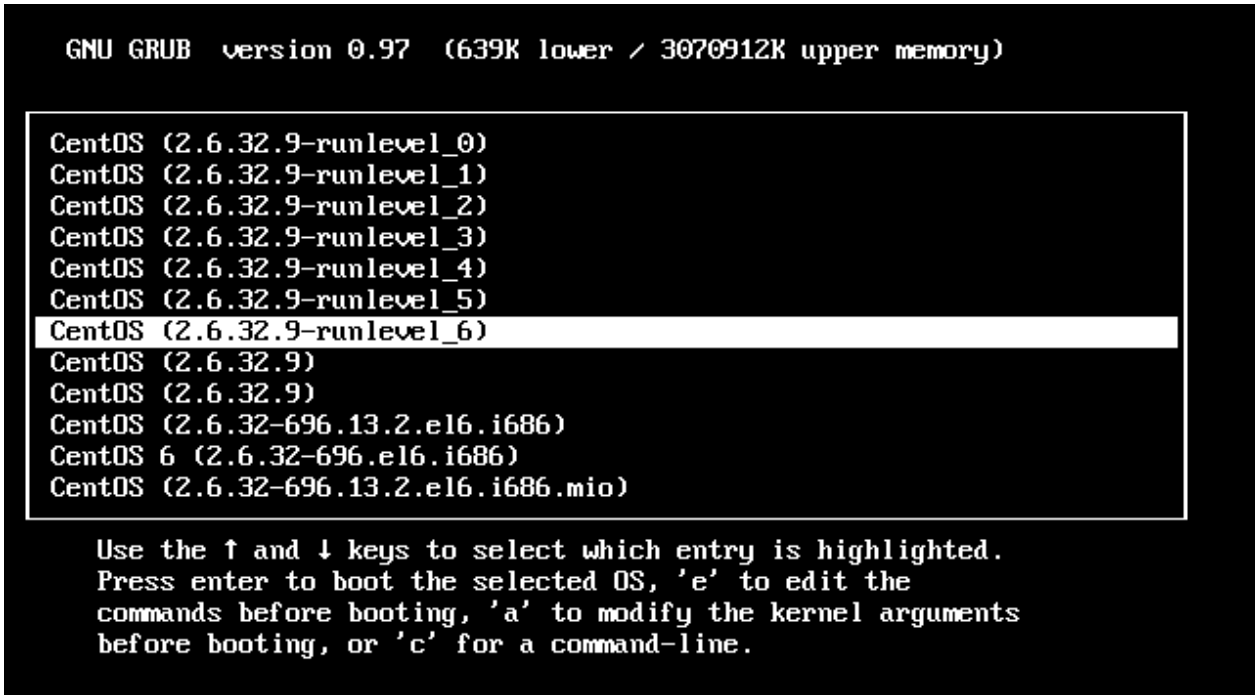
- **Nivel de arranque 5: Multiusuario gráfico**



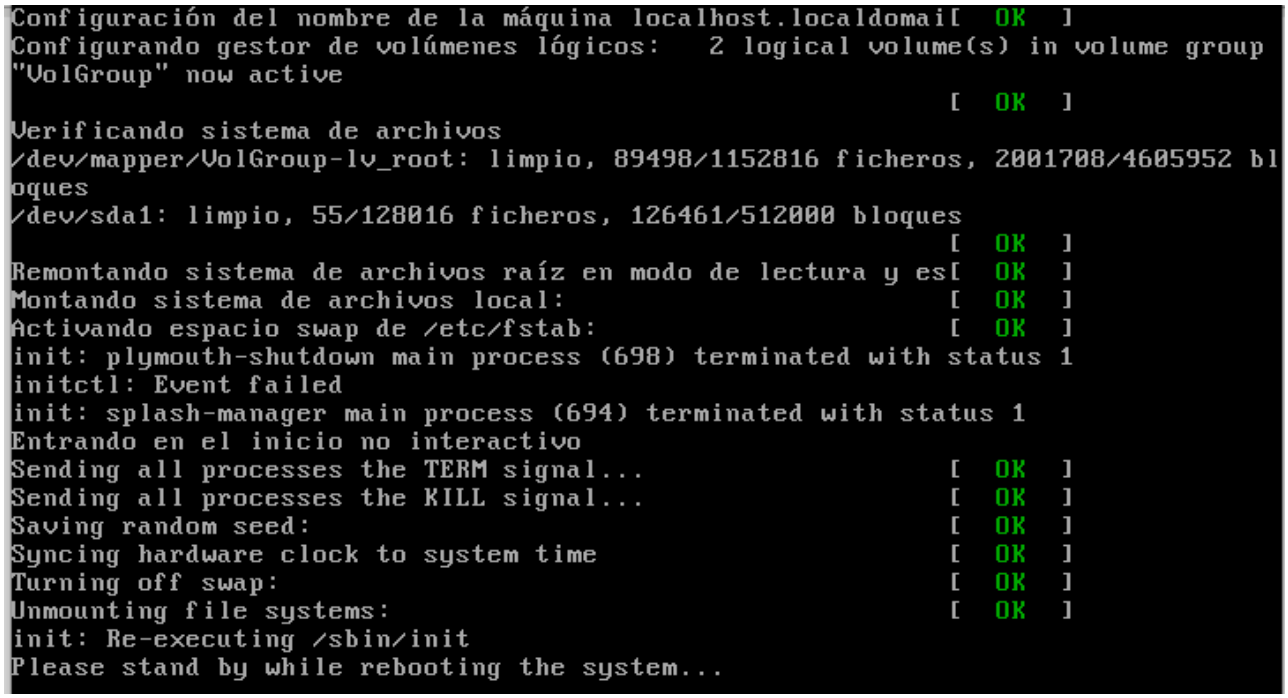
En el nivel de arranque 5, el sistema se cuelga al arrancar



- **Nivel de arranque 6: reinicio**



Vemos que, al iniciar el nivel 6, el sistema se reinicia nada más iniciarse.



4. Cambiando la clave de root desde GRUB

GRUB nos permite indicar en la línea de arranque del kernel que programa queremos ejecutar. Esto se hace con el argumento *init,* de sintaxis:

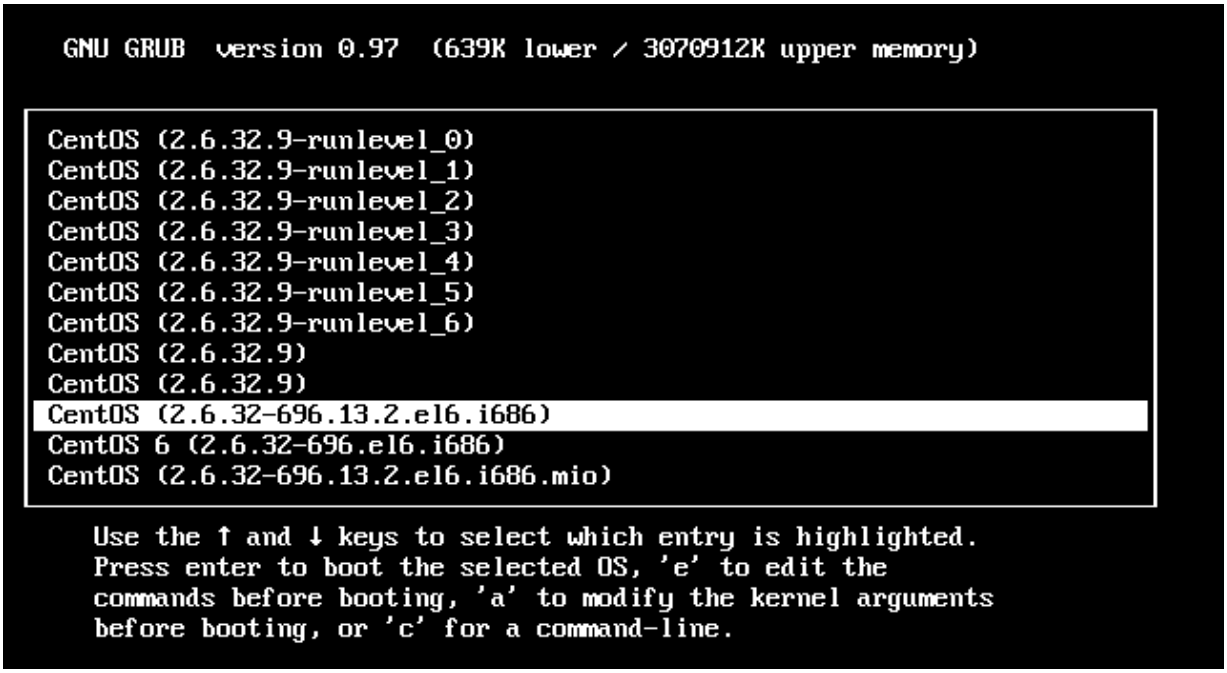
init=[ruta_programa]

Si el programa que ponemos a ejecutar es el interprete de comandos, podremos tener acceso al sistema y, entre otras cosas, podremos cambiar la clave de root.

Para ello, seguimos el siguiente proceso:

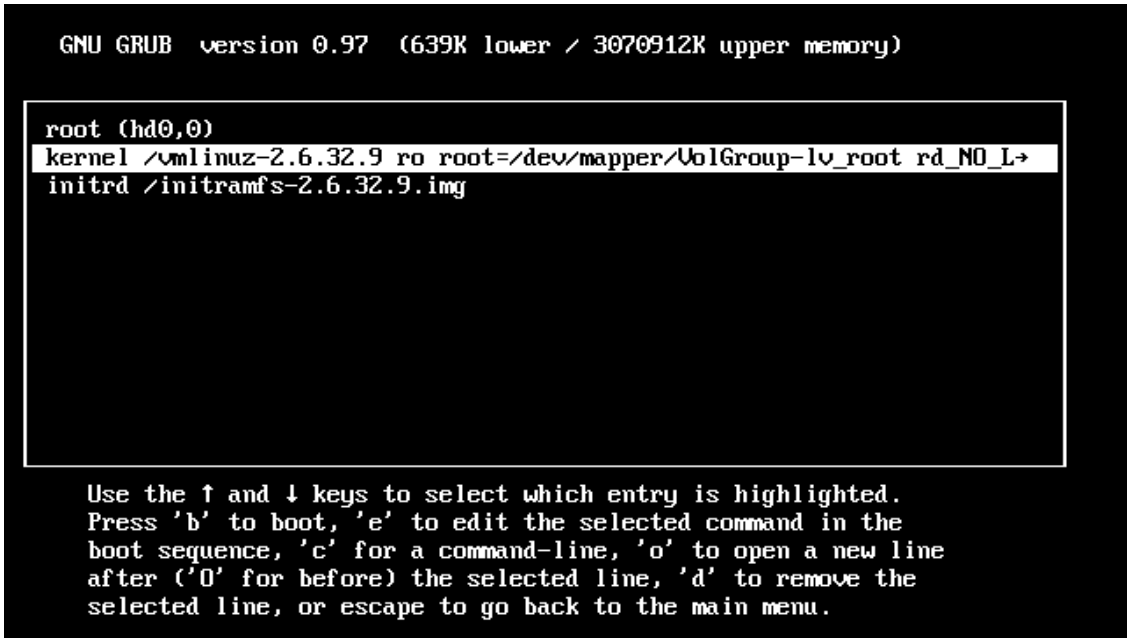
1. Iniciamos la máquina y abrimos el menú de GRUB

Antes de arrancar el sistema, si pulsamos cualquier tecla, se nos abrirá el menú principal del GRUB, similar a este.



2. Entramos a la configuración de nuestra entrada del kernel

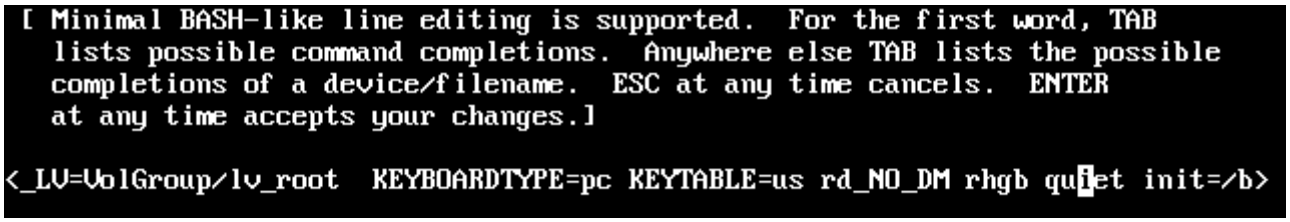
El el menú del GRUB, nos posicionamos en la entrada que queremos editar, y pulsamos ‘e’.
Esto nos abrirá la siguiente pantalla, con los datos del *grub.cfg* correspondientes a dicha entrada



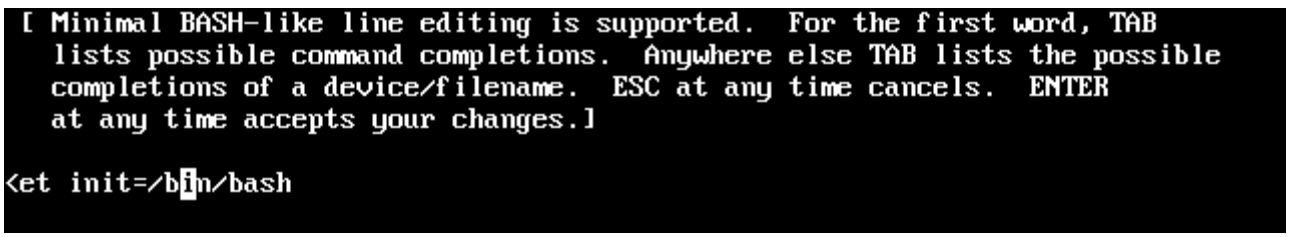
3. Editamos el campo *kernel,* añadiendo la opción *init*

Pulsamos ‘e’, para editar el campo *kernel.*

Nos aparecerá una pantalla como esta:



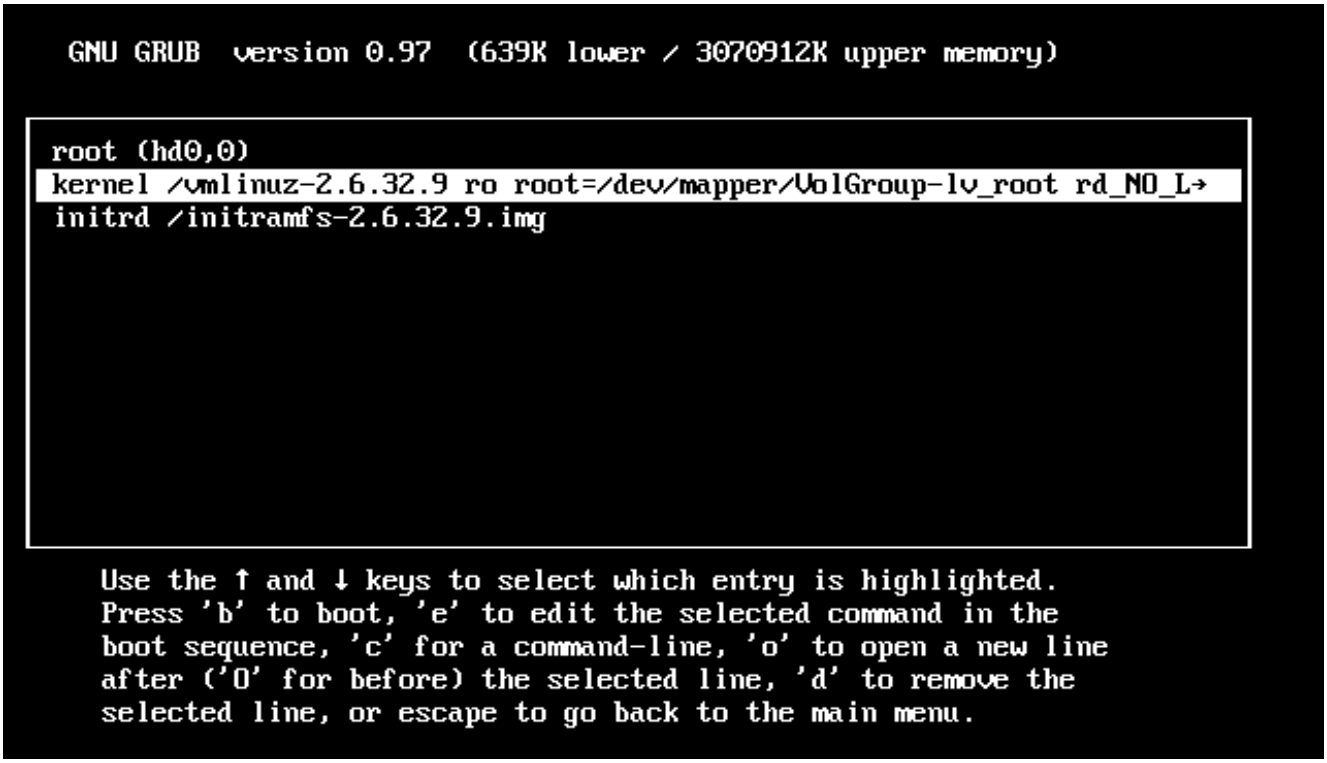
Nos vamos al final de la línea y añadimos el argumento: *init=/bin/bash*



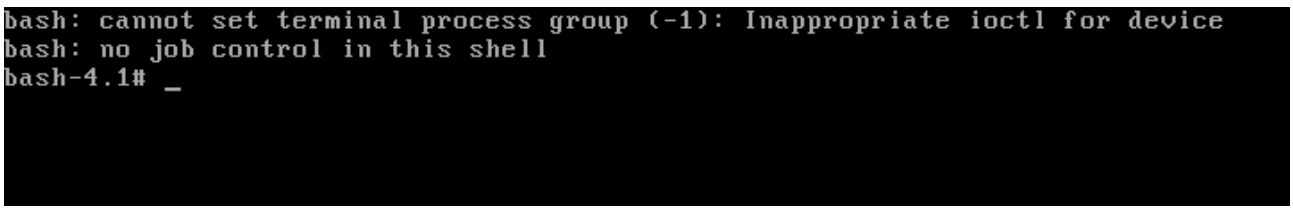
Pulsamos enter para volver a la pantalla anterior

4. Arrancamos el kernel

Pulsamos ‘b’ para iniciar el kernel con la nueva configuración



Esto nos abrirá un interprete de comandos similar a éste:

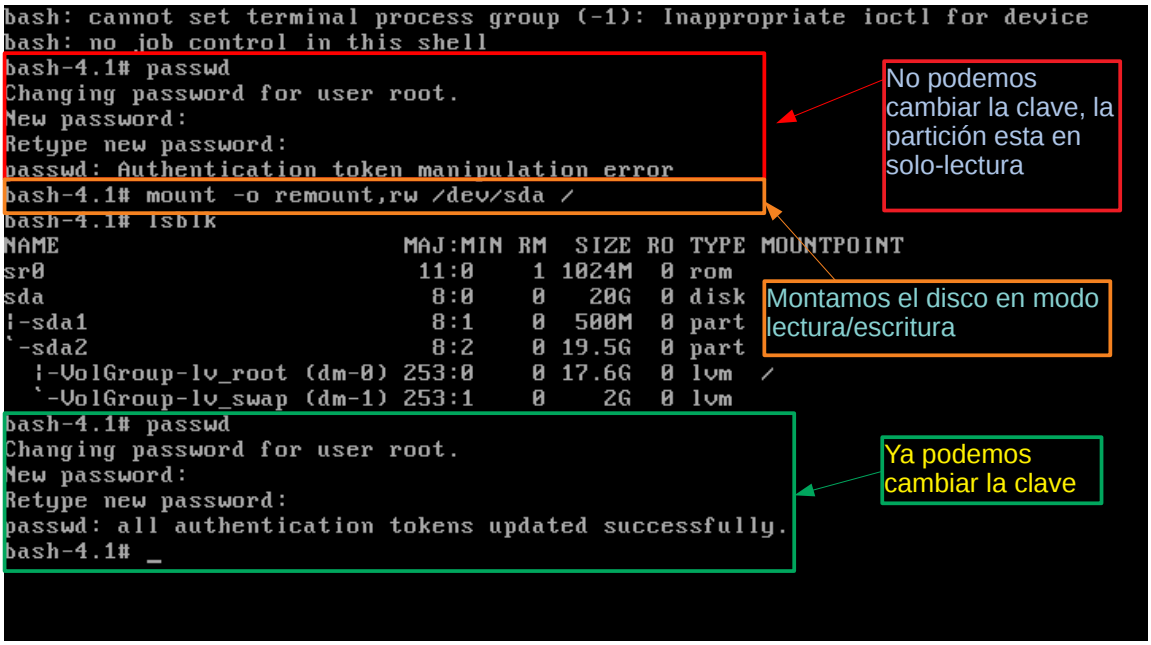


5. Montamos de nuevo la partición en modo lectura/escritura

En este interprete de comandos, podremos ejecutar el comando *passwd,* que nos permitirá cambiar la clave de root.

Pero, con el arranque por defecto, la partición raiz esta montada en modo solo-lectura, por lo cual el comando nos producirá un error y nos impedirá cambiar la clave.

Montando de nuevo la partición en modo lectura/escritura, tendremos el acceso que necesitamos para poder cambiar nuestra clave de usuario.



6. Conclusiones

La personalización del gestor de arranque y de los niveles de arranque del init nos permiten tener un control muy preciso de la forma en que se inicia nuestro sistema, y los recursos de los que dispondrá.

GRUB nos aporta una gran flexibilidad en sus entradas del kernel, lo cual nos permite tareas tales como cambiar una clave del usuario sin tener que iniciar el sistema, indicar el nivel de arranque en que queremos iniciar, pasarle parámetros al kernel… todo de forma relativamente simple, editando las entradas del menú.

Por otra parte, sysvinit nos aporta una gestión del arranque de nuestro sistema, en forma de niveles de arranque, muy simple y flexible, aunque algo tediosa de gestionar en algunos casos.