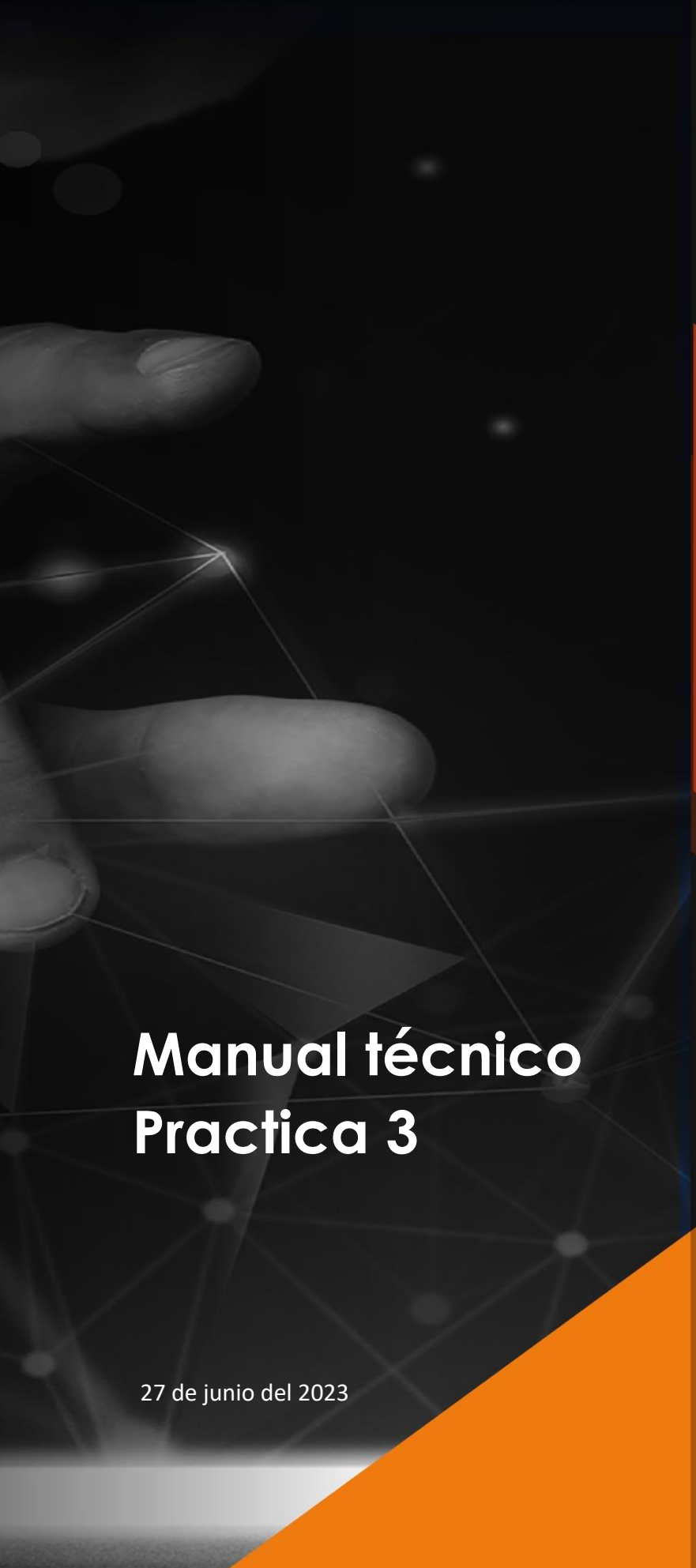


Manual técnico
Practica 3

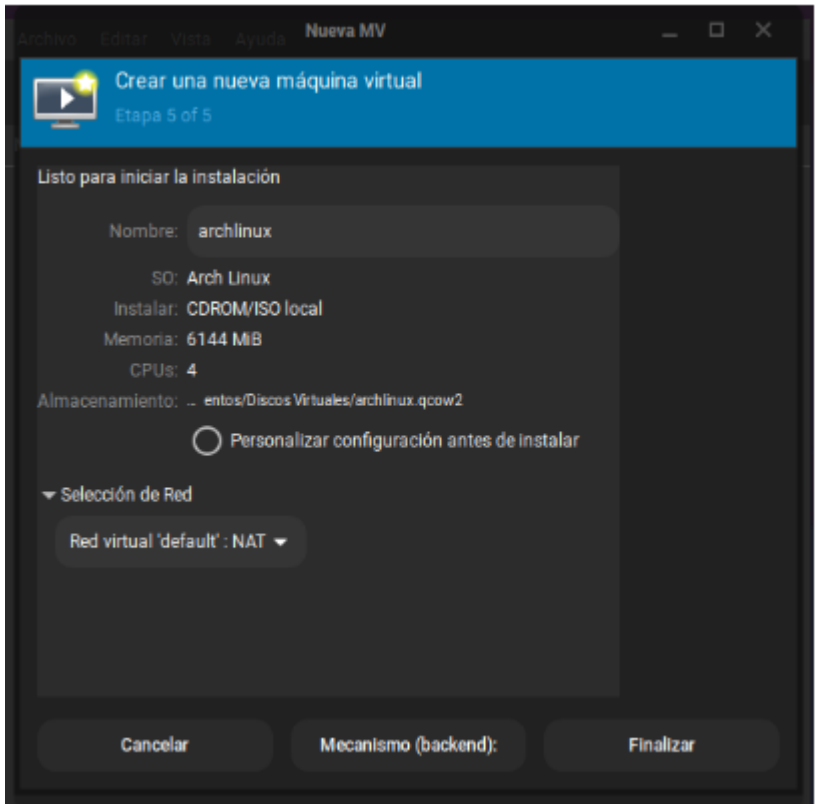
27 de junio del 2023

Grupo 2

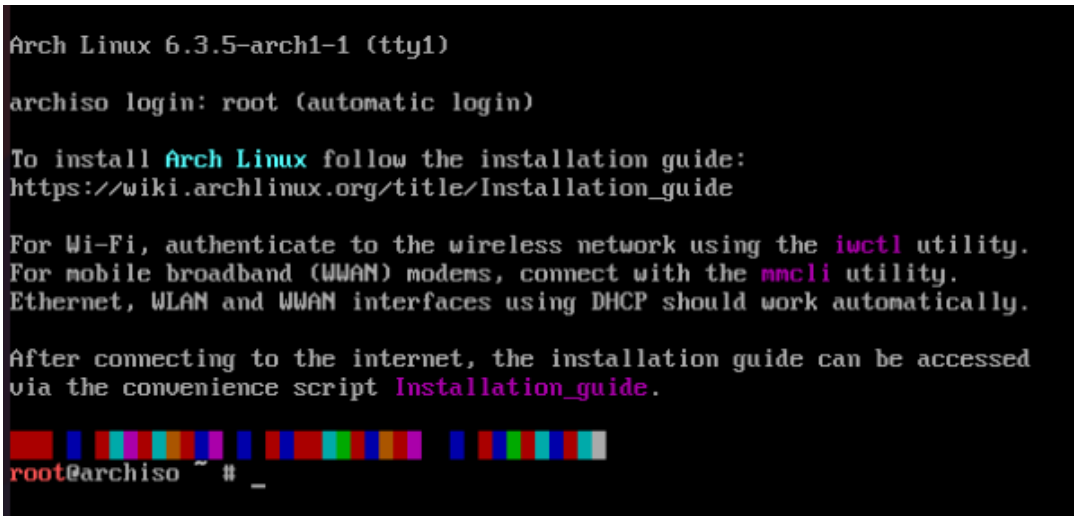
Arch Linux

MAQUINA VIRTUAL

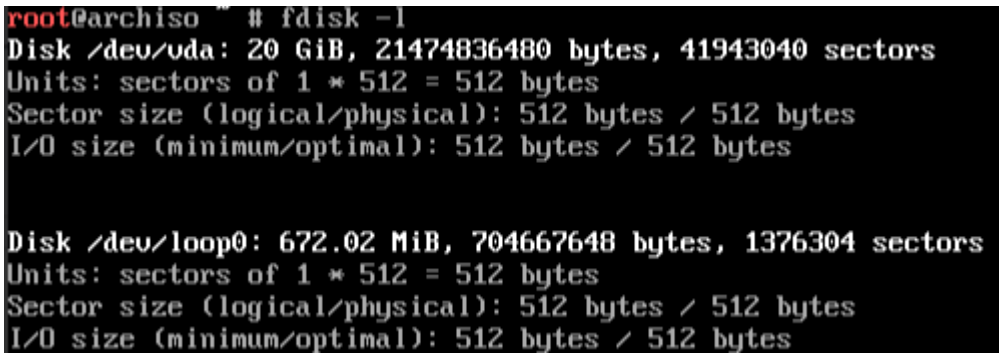
- Software de virtualización: Gestor de máquinas virtuales
- Nombre: archlinux
- Imagen ISO: https://archlinux.org/download/
- Memoria asignada: 6144 MB
- CPU: 4 Cores
- Almacenamiento: 20gb
- Red: NAT default



INSTRUCCIONES DE INSTALACION



1. Creación de las particiones
    a. Listar discos: “fdisk -l”



    b. Iniciar herramienta de particionamiento de discos: “fdisk /dev/vda”

c. Creación de partición SWAP: “n”
   i. Tipo: p
   ii. Number: 1 Default
   iii. First Sector: Default
   iv. Last Secor: +4G

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-41943039, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-41943039, default 41943039): +4G

Created a new partition 1 of type 'Linux' and of size 4 GiB.
```

d. Creación de partición ROOT: “n”
   i. Tipo: p
   ii. Number: 2 Default
   iii. First Sector: Default
   iv. Last Secor: Default

```
Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2):
First sector (8390656-41943039, default 8390656):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (8390656-41943039, default 41943039):

Created a new partition 2 of type 'Linux' and of size 16 GiB.
```

e. Salir y guardar los cambios: “w”

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

2. Formatear Particiones
    a. SWAP: “mkswap /dev/vda1”

```
root@archiso ~ # mkswap /dev/vda1
Setting up swapspace version 1, size = 4 GiB (4294963200 bytes)
no label, UUID=514d0a5e-e1b4-495b-93ea-68136ae686a2
```

    b. ROOT: “mkfs.ext4 /dev/vda2”

```
root@archiso ~ # mkfs.ext4 /dev/vda2
mke2fs 1.47.0 (5-Feb-2023)
Discarding device blocks: done
Creating filesystem with 4194048 4k blocks and 1048576 inodes
Filesystem UUID: 3bb4be03-9bb5-4d05-b76f-a89f013cf89a
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

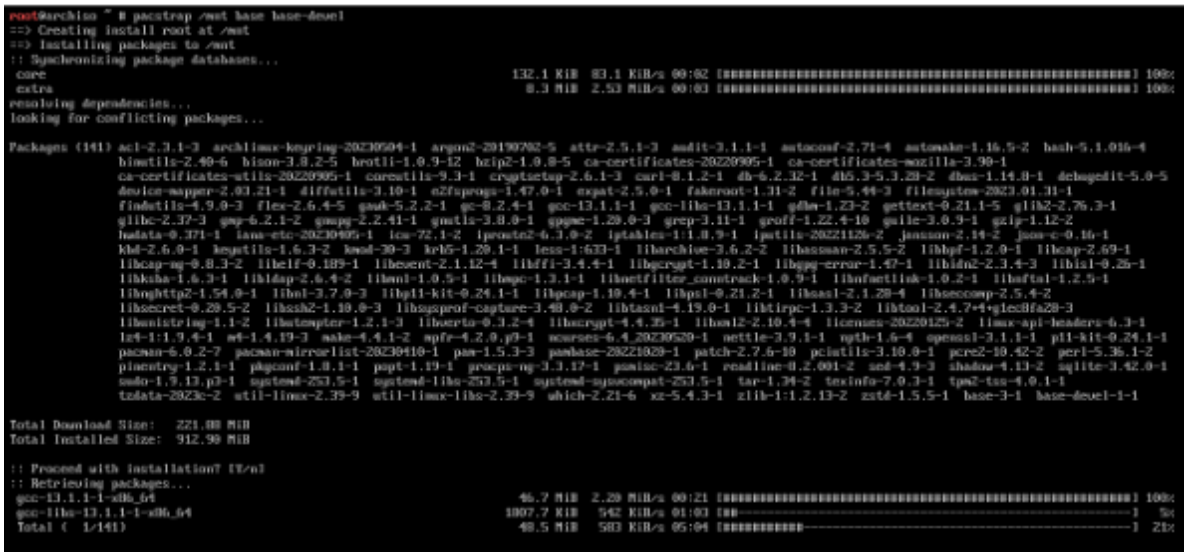
3. Montar particiones
    a. Partición root: “mount /dev/vda2 /mnt

```
root@archiso ~ # mount /dev/vda2 /mnt
```

    b. Particion swap: “swapon /dev/vda1”

```
root@archiso ~ # swapon /dev/vda1
```

4. Instalación
    “pacstrap /mnt base base-devel”



Configuración del sistema

1. Fstab: “genfstab -U /mnt >> /mnt/etc/fstab”

```
root@archiso ~ # genfstab -U /mnt >> /mnt/etc/fstab
```

2. Chroot: “arch-chroot /mnt”

```
root@archiso ~ # arch-chroot /mnt
[root@archiso /]#
```

3. Zona horaria: “ln -sf /usr/share/zoneinfo/Mexico/BajacSur /etc/localtime”

```
[root@archiso /]# ln -sf /usr/share/zoneinfo/Mexico/BajacSur /etc/localtime
```

4. Instalar Nano: “pacman -S nano”

```
[root@archiso /]# pacman -S nano
resolving dependencies...
looking for conflicting packages...

Packages (1) nano-7.2-1

Total Download Size:   0.58 MiB
Total Installed Size:  2.51 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
 nano-7.2-1-x86_64                  598.6 KiB   299 KiB/s 00:02 [##########################################################] 100%
(1/1) checking keys in keyring                                  [##########################################################] 100%
(1/1) checking package integrity                                [##########################################################] 100%
(1/1) loading package files                                     [##########################################################] 100%
(1/1) checking for file conflicts                               [##########################################################] 100%
(1/1) checking available disk space                             [##########################################################] 100%
:: Processing package changes...
(1/1) installing nano                                           [##########################################################] 100%
:: Running post-transaction hooks...
(1/2) Arming ConditionNeedsUpdate...
(2/2) Updating the info directory file...
```

5. Instalación del Kernel:
    1. Agregar nuevo repositorio a pacman
        i. “nano /etc/pacman.conf”
        ii. Agregar la siguiente información al archivo:

```
[kernel-lts]
Server=https://repo.m2x.dev/current/$repo/$arch
```

    2. Firmar repositorio:
        i. “pacman-key –recv-keys 76C6E477042BFE985CC220BD9C08A255442FAFF0”

```
[root@archiso /]# pacman-key --recv-keys 76C6E477042BFE985CC220BD9C08A255442FAFF0
warning: config file /etc/pacman.conf, line 100: directive 'server' in section 'kernel-lts' not recognized.
gpg: key 9C08A255442FAFF0: public key "Jonathon Fernyhough <jonathon@m2x.dev>" imported
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   5  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1  valid:   5  signed:  96  trust: 0-, 0q, 0n, 5m, 0f, 0u
gpg: depth: 2  valid:  73  signed:  27  trust: 73-, 0q, 0n, 0m, 0f, 0u
gpg: next trustdb check due at 2023-07-12
gpg: Total number processed: 1
gpg:               imported: 1
```

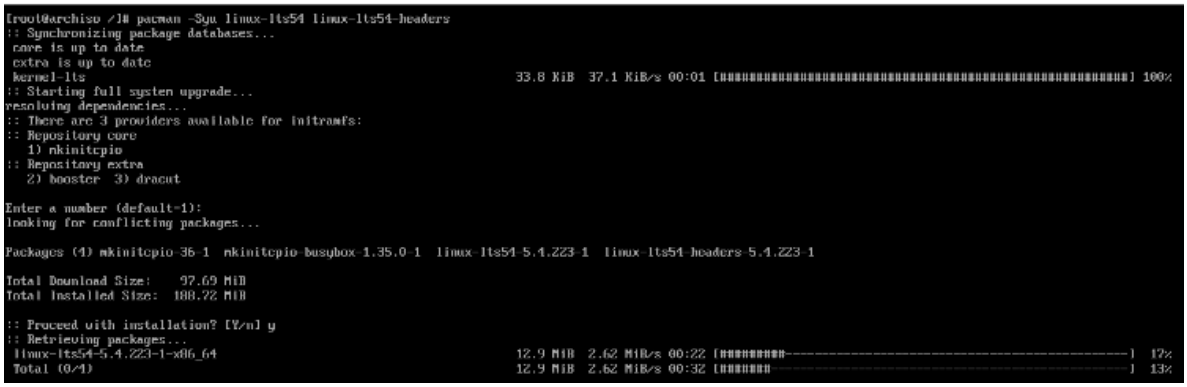
        ii. “pacman-key –finger 76C6E477042BFE985CC220BD9C08A255442FAFF0”

```
[root@archiso /]# pacman-key --finger 76C6E477042BFE985CC220BD9C08A255442FAFF0
pub   rsa4096 2014-03-31 [SCA]
      76C6 E477 042B FE98 5CC2  20BD 9C08 A255 442F AFF0
uid           [ unknown] Jonathon Fernyhough <jonathon@m2x.dev>
uid           [ unknown] Jonathon Fernyhough <jonathon@m2linux.com>
uid           [ unknown] Jonathon Fernyhough <jonathon@mange-tout.org>
uid           [ unknown] Jonathon Fernyhough <jonathon@manjaro.org>
sub   rsa4096 2014-03-31 [E]
```

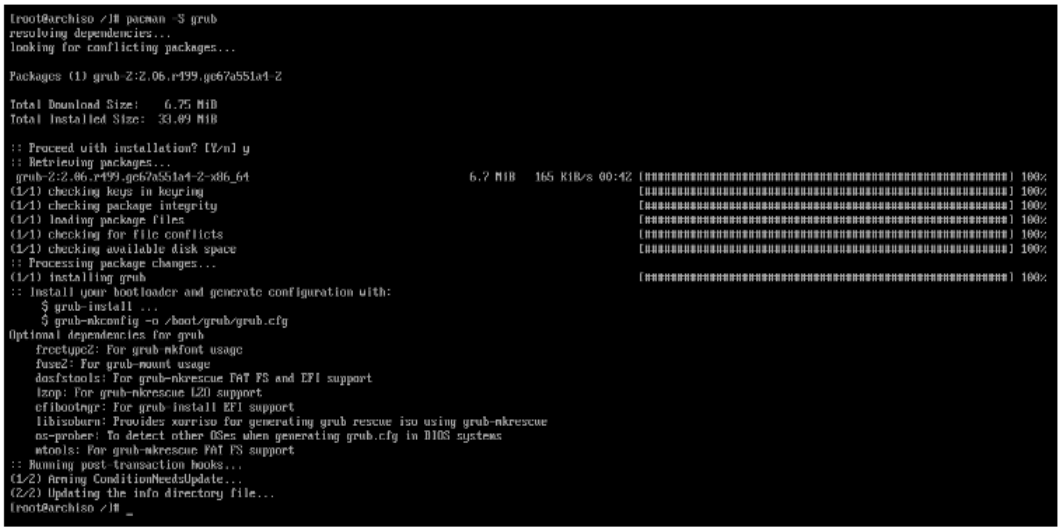
iii. “pacman-key –lsign-key 76C6E477042BFE985CC220BD9C08A255442FAFF0”

3. Instalar el kernel

“pacman -Syu Linux-lts54 linux-lts54-headers”



6. Instalación grub
    1. Descargar grub: “pacman -S grub”



2. Instalar grub: “grub-install /dev/vda

3. Generar archivo de configuración del grub: “grub-mkconfig -o /boot/grub/grub.cfg”

```
[root@archiso /]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux-lts54
Found initrd image: /boot/initramfs-linux-lts54.img
Found fallback initrd image(s) in /boot:  initramfs-linux-lts54-fallback.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

7. Agregar idiomas
    1. Ingresar archivo de idiomas: “nano /etc/locale.gen”

    ```
    [root@archiso /]# nano /etc/locale.gen
    ```

    2. Des comentar los idiomas deseados y guardar los cambios
    3. Generar los idiomas: “locale-gen”

    ```
    [root@archiso /]# locale-gen
    Generating locales...
      en_US.UTF-8... done
      es_GT.UTF-8... done
    Generation complete.
    ```

    4. Establecer el idioma principal: “echo LANG=es_GT.UTF-8 > /etc/locale.conf” “export LANG=es_GT.UTF-8”
8. Configurar red

- Establecer el hostname de la máquina.

```
[root@archiso /]# echo sopes2 > /etc/hostname
```

- Crear archivo de hosts y editar su configuración.

```
[root@archiso /]# touch /etc/hosts

[root@archiso /]# nano /etc/hosts
```

- Configuración dentro del archivo de hosts.

```
# Static table lookup for hostnames.
# See hosts(5) for details.
127.0.0.1       localhost
127.0.1.1       sopes2
::1             localhost
```

- Crear una contraseña al usuario root.

```
[root@archiso /]# passwd
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
```

- Crear un nuevo usuario, asignarle una contraseña y asignarle grupos.

```
[root@archiso /]# useradd -m so2_practica2_2
[root@archiso /]# passwd so2_practica2_2
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
```

```
[root@archiso /]# usermod -aG wheel,audio,video,storage so2_practica2_2
```

- Permitir que los miembros del grupo wheel puedan ejecutar cualquier comando.

- Editar el archivo visudo.

```
[root@archiso /]# EDITOR=nano visudo
```

- Descomentar la siguiente línea y guardar los cambios.

```
## Uncomment to allow members of group wheel to execute any command
_%wheel ALL=(ALL:ALL) ALL
```

9. Instalación de interfaz grafica
    1. Instalar xorg y networkmanager: “pacman -S xorg networkmanager”

```
[root@archiso /]# pacman -S xorg networkmanager
```

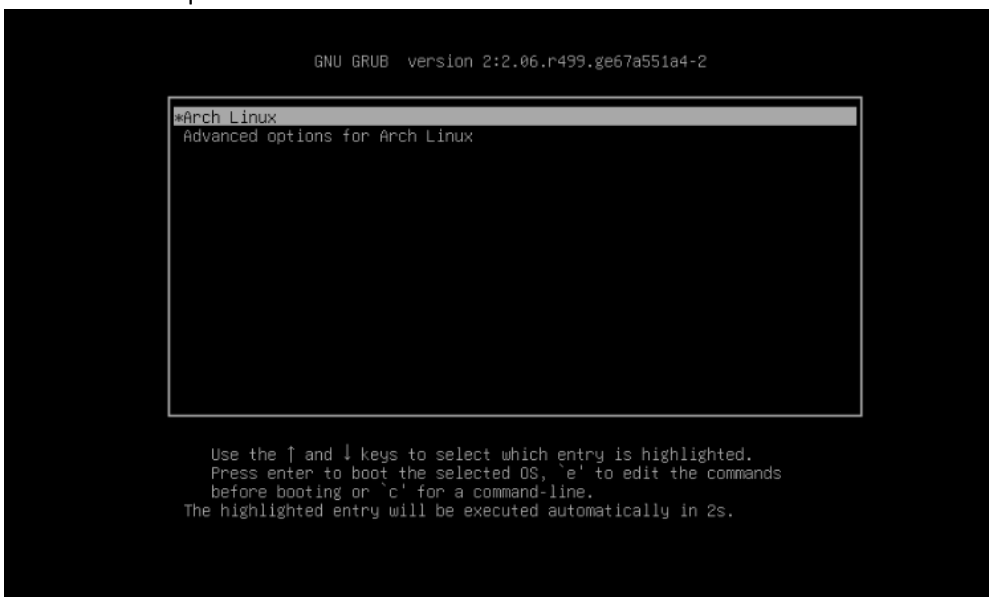
    2. Instalar gnome: “pacman -S gnome”

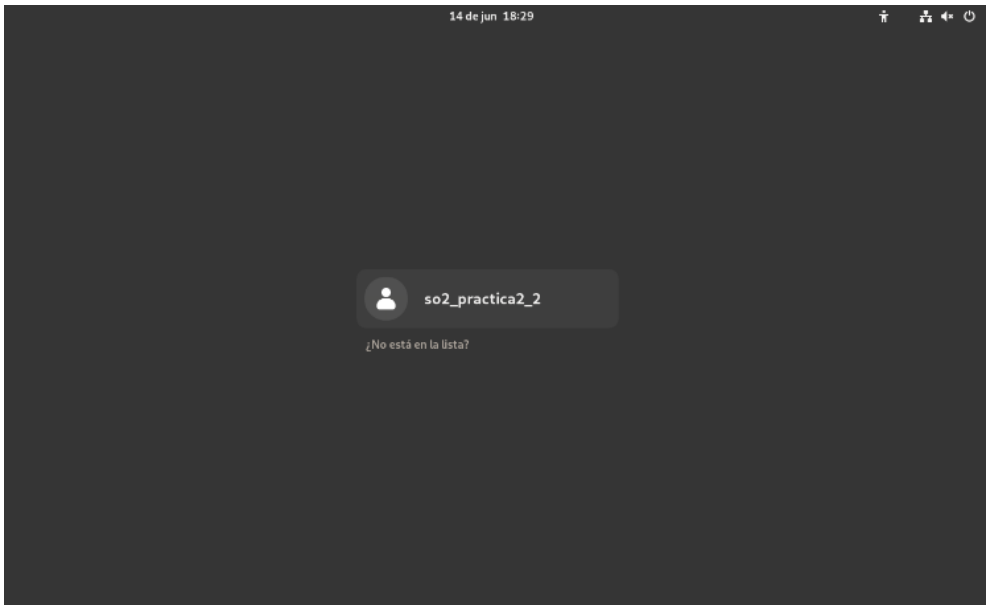
```
[root@archiso /]# pacman -S gnome
```

    3. Inicializar y habilitar servicios de interfaz grafica

```
[root@archiso /]# systemctl enable gdm.service
Created symlink /etc/systemd/system/display-manager.service → /usr/lib/systemd/system/gdm.service.
[root@archiso /]# systemctl enable NetworkManager.service
Created symlink /etc/systemd/system/multi-user.target.wants/NetworkManager.service → /usr/lib/systemd/system/NetworkManager.service.
Created symlink /etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service → /usr/lib/systemd/system/NetworkManager-dispatcher.service.
Created symlink /etc/systemd/system/network-online.target.wants/NetworkManager-wait-online.service → /usr/lib/systemd/system/NetworkManager-wait-online.service.
```

4. Reiniciar la maquina virtual





Cambios realizados a los módulos

MODULO men_grupo2

1. Uso de file_operations en lugar de proc_ops debido a la versión del kernel que se está utilizando.

Antes:

```
static struct proc_ops operations = {
    .proc_open = read,
    .proc_read = seq_read};
```

Después:

```
static struct file_operations operations =
    {
        .open = read,
        .read = seq_read};
```

MODULO cpu_grupo2

1. Uso de file_operations en lugar de proc_ops debido a la versión del kernel que se está utilizando.

Antes:

```
static struct proc_ops operations = {
    .proc_open = read,
    .proc_read = seq_read};
```

Después:

```
static struct file_operations operations =
    {
        .open = read,
        .read = seq_read};
```

2. Importación de librería para el uso de la estructura task_struct.

Antes:

```
#include <linux/sched.h>
```

Después:

```
#include <linux/sched/signal.h>
```

3. Obtención del estado de un proceso.

Antes:

```
process->__state
```

Después:

```
process->state
```

Software Utilizado

npm

- Comando de instalación
  `sudo pacman -S npm`

localtunel

- Comando de instalación
  `npm install -g localtunnel`

- Comando de ejecución
  `lt –port 8080`

- Obtener dirección ip para acceder a la url generada
  `curl ipv4.icanhazip.com`

Gorila Mux

- Comando de instalación

```
go get -u github.com/gorilla/mux
```

MODULOS

MODULO DE RAM

Este modulo de kernel de Linux proporciona información sobre la memoria RAM del sistema.

Librerías

- **linux/module.h:** Permite la definición de módulos del kernel de Linux. Proporciona funciones para la carga y descarga de los módulos.
- **Linux/kernel.h:** Esta librería permite interactuar con el núcleo del sistema operativo por ejemplo la impresión de mensajes en el registro de Kernel.
- **Linux/mm.h:** Esta librería tiene funciones y estructuras relacionadas con la administración de memoria del kernel de Linux.
- **Linux/init.h:** Esta librería tiene funciones para la inicialización y finalización del módulo.
- **Linux/proc_fs.h:** Esta librería permite trabajar con el sistema de archivos /proc en Linux.
- **Asm/uaccess.h:** Permite copiar datos entre el espacio de usuario y el espacio del kernel.
- **Linux/seq_file.h:** Esta librería proporciona funciones para trabajar con el sistema de archivos /proc y poder escribir en el archivo y generar una salida con el comando cat.

Código:

```
// Constantes de utilidad
const long MEGABYTE = 1024 * 1024;

// Variables para almacenar estadísticas del sistema
struct sysinfo sysinfo;
unsigned long totalRam;
unsigned long freeRam;
unsigned long occupiedRam;
unsigned long percentageOccupiedRam;
```

Constantes y variables en donde se guardará la información de la memoria RAM del sistema.

```
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Modulo de RAM");
MODULE_AUTHOR("Grupo 2");
```

Credenciales del módulo.

```c
static void init_meminfo(void)
{
    si_meminfo(&sysinfo);
}
```

Esta función inicializa la estructura sysinfo con la información actualizada sobre la memoria del sistema utilizando la función si_meminfo de la librería Linux/mm.h

```c
static int write(struct seq_file *file, void *v)
{
    init_meminfo();
    totalRam = ((sysinfo.totalram * sysinfo.mem_unit) / MEGABYTE);
    freeRam = ((sysinfo.freeram * sysinfo.mem_unit) / MEGABYTE);
    occupiedRam = totalRam - freeRam;
    percentageOccupiedRam = ((occupiedRam * 100) / totalRam);
    seq_printf(file, "{ \"Total\": %li , \"Occupied\": %li , \"Percentage\": %li , \"Free\": %l
    return 0;
}
```

Esta función se utiliza para escribir los datos de la memoria ram dentro del archivo generado por el módulo en /proc.

```c
static int read(struct inode *inode, struct file *file)
{
    return single_open(file, write, NULL);
}
```

Esta función se utiliza para leer los datos del archivo que genera el modulo en /proc y devuelve los resultados en single_open.

```c
static struct proc_ops operations = {
    .proc_open = read,
    .proc_read = seq_read};
```

Esta estructura define las operaciones que se realizaran en el archivo virtual que genera el módulo. Especifica las funciones que se llamaran cuando se abra el modulo y cuando se lea el archivo.

```c
static int _insert(void)
{
    proc_create("mem_grupo2", 0, NULL, &operations);
    printk(KERN_INFO "Hola mundo, somos el grupo 2 y este es el monitor de memoria.\n");
    return 0;
}
```

Esta función se ejecuta cuando se inserta este módulo de memoria en el kernel de Linux con el comando insmod. Crea una entrada en el sistema de archivos /proc para el archivo “mem_grupo2” y también imprime un mensaje con el comando printk.

```
static void _remove(void)
{
    remove_proc_entry("mem_grupo2", NULL);
    printk(KERN_INFO "Sayonara mundo, somos el grupo 2 y este fue el monitor de memoria.\n");
}
```

Esta función se ejecuta cuando se elimina el modulo de memoria del kernel de Linux utilizando el comando rmod. Elimina la entrada del sistema de archivos /proc para el archivo “mem_grupo2” y también imprime un mensaje con el comando printk.

```
module_init(_insert);
module_exit(_remove);
```

Asigna los métodos que se utilizaran para cuando se inserta o elimina el modulo de memoria dentro del kernel de Linux.

MODULO DE CPU

Este módulo de kernel de Linux proporciona información sobre los procesos en ejecución y el uso del CPU.

Librerías:

- **linux/module.h:** Permite la definición de módulos del kernel de Linux. Proporciona funciones para la carga y descarga de los módulos.
- **Linux/kernel.h:** Esta librería permite interactuar con el núcleo del sistema operativo por ejemplo la impresión de mensajes en el registro de Kernel.
- **Linux/mm.h:** Esta librería tiene funciones y estructuras relacionadas con la administración de memoria del kernel de Linux.
- **Linux/init.h:** Esta librería tiene funciones para la inicialización y finalización del módulo.
- **Linux/proc_fs.h:** Esta librería permite trabajar con el sistema de archivos /proc en Linux.
- **Linux/seq_file.h:** Esta librería proporciona funciones para trabajar con el sistema de archivos /proc y poder escribir en el archivo y generar una salida con el comando cat.
- **Linux/sched/signal.h:** Esta librería contiene definiciones relacionadas para los struct de task

Código:

Variables para almacenar información de los procesos del CPU.

```
struct task_struct *process;
struct task_struct *task_child;
struct list_head *list;
```

Credenciales del módulo.

```
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Modulo de CPU");
MODULE_AUTHOR("Grupo 2");
```

Función write:

Esta función se llama cuando se lee el archivo del módulo. Recorre todos los procesos en ejecución y recopila información sobre cada proceso luego escribe estos datos recopilados con la función seq_printf().

```
static int write(struct seq_file *file, void *v)
{
    long rss;
    bool esPadre, first, next = true, conhijos;

    seq_printf(file, "{\"root\": [\n");
    for_each_process(process)
    {
        first = true;
        if (process->mm)
        {
            rss = get_mm_rss(process->mm);

            if (next)
            {
                seq_printf(file, "{\"Process\":\"%s\",\n\"PID\":\"%d\",\n\"RAM\":\"%ld
                next = false;
            }
            else
            {
                seq_printf(file, ",{\"Process\":\"%s\",\n\"PID\":\"%d\",\n\"RAM\":\"%l
            }
            esPadre = true;
        }
        else
        {
            if (next)
            {
                seq_printf(file, "{\"Process\":\"%s\",\n\"PID\":\"%d\",\n\"RAM\":\"0\"
                next = false;
            }
            else
            {
                seq_printf(file, ",{\"Process\":\"%s\",\n\"PID\":\"%d\",\n\"RAM\":\"0\
```

Función read: Esta función se utiliza para leer los datos del archivo que genera el modulo en /proc y devuelve los resultados en single_open.

```
static int read(struct inode *inode, struct file *file)
{
    return single_open(file, write, NULL);
}
```

Esta estructura define las operaciones que se realizaran en el archivo virtual que genera el módulo. Especifica las funciones que se llamaran cuando se abra el modulo y cuando se lea el archivo.

```
static struct proc_ops operations = {
    .proc_open = read,
    .proc_read = seq_read};
```

Esta función se ejecuta cuando se inserta este módulo de CPU en el kernel de Linux con el comando insmod. Crea una entrada en el sistema de archivos /proc para el archivo “cpu_grupo2” y también imprime un mensaje con el comando printk.

```
static int _insert(void)
{
    proc_create("cpu_grupo2", 0, NULL, &operations);
    printk(KERN_INFO "Hola mundo, somos el grupo 2 y este es el monitor de CPU.\n");
    return 0;
}
```

Esta función se ejecuta cuando se elimina el módulo de CPU del kernel de Linux utilizando el comando rmod. Elimina la entrada del sistema de archivos /proc para el archivo “cpu_grupo2” y también imprime un mensaje con el comando printk.

```
static void _remove(void)
{
    remove_proc_entry("cpu_grupo2", NULL);
    printk(KERN_INFO "Sayonara mundo, somos el grupo 2 y este fue el monitor de CPU.\n");
}
```

Asigna los métodos que se utilizaran para cuando se inserta o elimina el módulo de memoria dentro del kernel de Linux.

```
module_init(_insert);
module_exit(_remove);
```

BACKEND GO

Se desarrollo una API con GO que ejecuta el comando CAT para cada uno de los módulos descritos anteriormente y luego extrae la información de los archivos de los módulos en /proc para retornarlos al cliente.

```
import (
    "log"
    "net/http"
    cpuController "so2_practica1_2/backend/controllers/cpu"
    memoryAssignmentController "so2_practica1_2/backend/controllers/memory"
    ramController "so2_practica1_2/backend/controllers/ram"

    "github.com/gorilla/handlers"
    "github.com/gorilla/mux"
)
```

Este bloque importa las librerías necesarias para el funcionamiento del servidor web, incluyendo el framework Gin, las librerías estándar de Go y las definiciones de estructuras personalizadas.

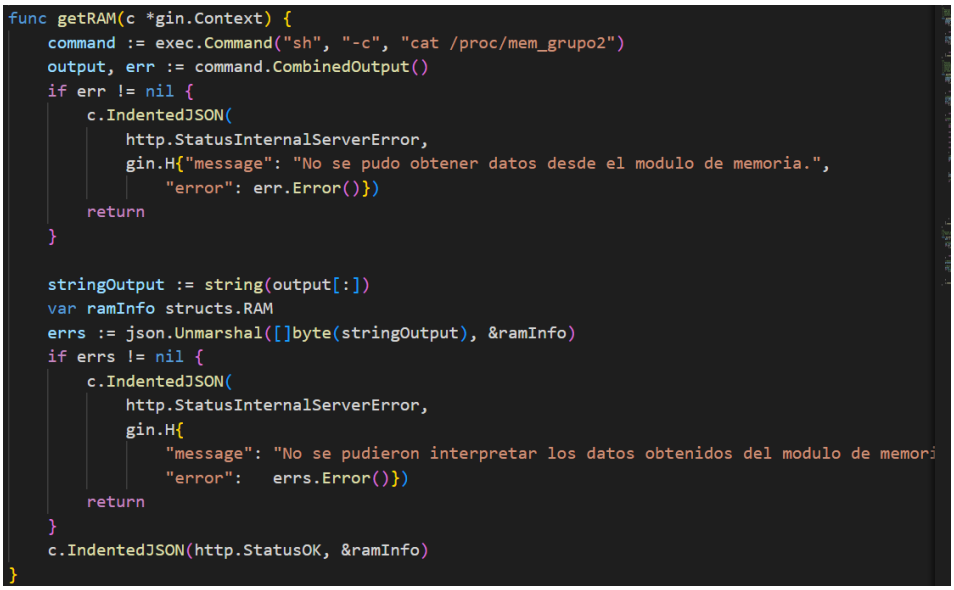
```
func main() {
    port := "8080"
    r := mux.NewRouter().StrictSlash(true)
    r.HandleFunc("/ram", ramController.GetRAM).Methods("GET")
    r.HandleFunc("/cpu", cpuController.GetCPU).Methods("GET")
    r.HandleFunc("/memory-assignment/{id}", memoryAssignmentController.GetMemoryAssignment).Methods("GET")
    r.HandleFunc("/kill/{id}", cpuController.KillProcess).Methods("DELETE")

    headersOk := handlers.AllowedHeaders([]string{"X-Requested-With", "Content-Type", "Authorization"})
    methodsOk := handlers.AllowedMethods([]string{"GET", "HEAD", "POST", "PUT", "OPTIONS", "DELETE"})
    originsOk := handlers.AllowedOrigins([]string{"*"})

    log.Println("Servidor iniciado en el puerto: " + port)
    error := http.ListenAndServe(":"+port, handlers.CORS(originsOk, headersOk, methodsOk)(r))

    if error != nil {
        log.Fatal("Error al iniciar el servidor.\n", error)
    }

}
```

La función main() es el punto de entrada del programa. Aquí se configura el enrutador de Gin y se definen las rutas de la API:

- GET /ram: Invoca la función getRAM para obtener información sobre la memoria RAM del servidor.
- GET /cpu: Invoca la función getCPU para obtener información sobre los procesos en ejecución en el servidor.

- DELETE /kill/:id: Invoca la función killProcess para eliminar un proceso en base al ID proporcionado.
- GET /memory-assigment/{id}: Obtiene la información de asignación de memoria

La función getRAM maneja la ruta GET /ram. Ejecuta el comando cat /proc/mem_grupo2 para obtener información sobre la memoria RAM del servidor desde un archivo virtual en el sistema de archivos /proc. Luego, convierte los datos obtenidos en formato JSON a una estructura RAM definida en structs.RAM y responde con los datos en formato JSON.



```go
func getRAM(c *gin.Context) {
    command := exec.Command("sh", "-c", "cat /proc/mem_grupo2")
    output, err := command.CombinedOutput()
    if err != nil {
        c.IndentedJSON(
            http.StatusInternalServerError,
            gin.H{"message": "No se pudo obtener datos desde el modulo de memoria.",
                "error": err.Error()})
        return
    }

    stringOutput := string(output[:])
    var ramInfo structs.RAM
    errs := json.Unmarshal([]byte(stringOutput), &ramInfo)
    if errs != nil {
        c.IndentedJSON(
            http.StatusInternalServerError,
            gin.H{
                "message": "No se pudieron interpretar los datos obtenidos del modulo de memori
                "error":   errs.Error()})
        return
    }
    c.IndentedJSON(http.StatusOK, &ramInfo)
}
```

La función getCPU maneja la ruta GET /cpu. Ejecuta el comando cat /proc/cpu_grupo2 para obtener información sobre los procesos y uso de CPU del servidor desde un archivo virtual en el sistema de archivos /proc. Luego, convierte los datos obtenidos en formato JSON a una estructura CPU definida en structs.CPU y responde con los datos en formato JSON.

```go
func getCPU(c *gin.Context) {
    //Obtener procesos
    command := exec.Command("sh", "-c", "cat /proc/cpu_grupo2")
    output, err := command.CombinedOutput()
    if err != nil {
        c.IndentedJSON(
            http.StatusInternalServerError,
            gin.H{
                "message": "No se pudo obtener datos desde el modulo de CPU.",
                "error":   err.Error()})
        return
    }

    //Convetir datos obtenidos a estructura Processes
    stringOutput := string(output[:])
    var processes structs.CPU
    errs := json.Unmarshal([]byte(stringOutput), &processes)
    if errs != nil {
        c.IndentedJSON(
            http.StatusInternalServerError,
            gin.H{
                "message": "No se pudieron interpretar los datos obtenidos del modulo de CPU"
                "error":   errs.Error()})
        return
```

La función killProcess es el controlador de la ruta DELETE /process/:id en el servidor. Su función principal es eliminar un proceso en función del ID proporcionado. Se utiliza la biblioteca os para la búsqueda y la finalización del proceso.

```go
func killProcess(c *gin.Context) {
    pid := c.Param("id")

    //Casteo a entero del id recibido
    id, err := strconv.Atoi(pid)
    if err != nil {
        c.IndentedJSON(
            http.StatusInternalServerError,
            gin.H{
                "message": "No se pudo realizar el casteo a entero del PID proporcionado"
                "error":   err.Error()})
        return
    }

    //Obtener el proceso
    process, _ := os.FindProcess(id)

    //Matar el proceso
    error := process.Signal(syscall.SIGKILL)
    if error != nil {
        c.IndentedJSON(
            http.StatusInternalServerError,
            gin.H{
                "message": "No se pudo matar el proceso.",
                "error":   err.Error()})
    }

    c.IndentedJSON(http.StatusOK, gin.H{"message": "Proceso Eliminado Exitosamente"})
```

La función GetMemoryAssignment es la función asignada a la ruta GET /memory-assigment/{id} y básicamente obtiene la información de asignación de memoria desde maps. Itera en cada línea y obtiene la información de cada columna, luego la interpreta y las almacena en un struct.

```go
func GetMemoryAssignment(w http.ResponseWriter, r *http.Request) {
    pid := mux.Vars(r)["id"]

    command := exec.Command("sh", "-c", "cat /proc/" + pid + "/maps")
    mapsData, catError := command.CombinedOutput()

    if catError != nil {
        log.Println("No se pudo obtener los datos de asginacion de memoria desde maps.\n" + catError.Error())
        commons.SendError(w, http.StatusInternalServerError)
        return
    }

    //Convetir de arreglo de bytes a arreglo de tipo string
    var lines []string = strings.Split(strings.ReplaceAll(string(mapsData[:]), "\r\n", "\n"), "\n")

    //Iniciar slice en donde se almacenaran los datos
    assignments := make([]memory.MEMORY, 0, len(lines))

    //Recorrer cada una de las lineas
    for i:=0; i<len(lines)-1; i++{

        //Obtener cada columna y setear el valor de la estructura MEMORY
        data := strings.Fields(lines[i])
        assignment := memory.MEMORY{
            Address: data[0],
            Size: data[2],
            Device: data[3],
        }

        //Interpretar y establecer los permisos
        var permisions string
        if(strings.Contains(data[1], "r")){
            permisions = permisions + "Lectura "
        }

        if(strings.Contains(data[1], "w")){
            permisions = permisions + "Escritura "
        }

        if(strings.Contains(data[1], "x")){
```

Practica 3 – Cambios backend GO

Se utiliza la librería encoding/json para deserializar los datos del modulo de memoria en la estructura ram.RAM y enviar estos datos en formato json.

```go
func GetRAM(w http.ResponseWriter, r *http.Request) {
    command := exec.Command("sh", "-c", "cat /proc/mem_grupo2")
    moduleData, catError := command.CombinedOutput()

    if catError != nil {
        log.Println("No se pudo obtener datos desde el modulo de memoria.\n" + catError.Error())
        commons.SendError(w, http.StatusInternalServerError)
        return
    }

    var ramData ram.RAM
    json.Unmarshal(moduleData, &ramData)
    jsonData, _ := json.Marshal(&ramData)

    commons.SendResponse(w, http.StatusOK, jsonData)
}
```

En el metodo GetMemoryAssignment después de obtener los datos del archivo maps estos se procesan y se almacenan en la estructura memory.MEMORY además se creó la función setRssAndSize para obtener más datos desde el archivo maps en este caso los datos Rss y Size de memoria en MB luego con estos datos se calcula la memoria residente y la memoria virtual. Por ultimo se serializan en formato json para retornarlo.

```go
        //Obtener los valores de RSS Y Size desde el archivo smaps
        if !setRssAndSize(&assignment, pid) {
            commons.SendError(w, http.StatusInternalServerError)
            return
        }

        residentMemory = residentMemory + assignment.Rss
        virtualMemory = virtualMemory + assignment.Size

        //Agregar la estructura actual al slice
        assignments.Assignments = append(assignments.Assignments, assignment)
    }

    assignments.ResidentMemory = math.Round(residentMemory*100) / 100
    assignments.VirtualMemory = math.Round(virtualMemory*100) / 100

    jsonData, _ := json.Marshal(assignments)
    commons.SendResponse(w, http.StatusOK, jsonData)
```

```go
/*
 * @apiNote: Método privado hace uso del comando cat para obtener los datos contenidos en el archivo smaps.
 * Filtra los datos utilizando el comando grep y obtiene unicamente los valores de Rss y Size. Convierte
 * el arreglo de bites obtenido en un arreglo de tipo string y obtiene los valores numéricos. Divide los
 * valores obtenidos entre 1024 para tenerlos en MB y los guarda en la estructura assignment.
 * @params assignment, puntero de la estructura donde se guardara la información
 *         pid, process id del proceso a buscar en la carpeta proc
 * return true, si todo se realizo con éxito
 *        false, si hubo un error obteniendo los datos.
 */
func setRssAndSize(assignment *memory.ASSIGNMENT, pid string) bool {
    command := exec.Command("sh", "-c", "cat /proc/"+pid+"/smaps | grep "+assignment.Address+" -A 4 | grep -E 'Rss|^Size'")
    data, catError := command.CombinedOutput()

    if catError != nil {
        log.Println("No se pudo obtener los datos de Rss y Size de memoria desde smaps.\n" + catError.Error())
        return false
    }

    var lines []string = strings.Split(strings.ReplaceAll(string(data[:]), "\r\n", "\n"), "\n")
    size, _ := strconv.ParseFloat(strings.Fields(lines[0])[1], 64)
    rss, _ := strconv.ParseFloat(strings.Fields(lines[1])[1], 64)

    assignment.Size = math.Round((size/1024)*100) / 100
    assignment.Rss = math.Round((rss/1024)*100) / 100

    return true
}
```