UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 1

SECCIÓN A

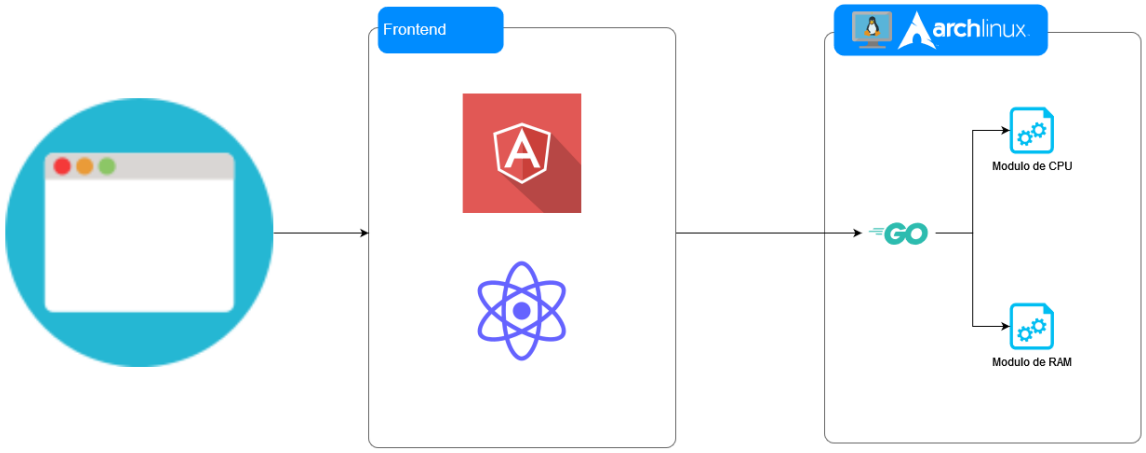


| Carne: | Nombre |
|---|---|
| 202004724 | Carlos Daniel Acabal Pérez |
| 202004765 | Javier Alejandro Gutierrez de León |
| 202000194 | Alvaro Emmanuel Socop Perez |


Guatemala, 18 de junio del 2023

## INTRODUCCIÓN

Se creó un sistema que se encarga de monitorear el uso en MB de memoria RAM, el porcentaje de CPU utilizado y los procesos que se encuentran activos, especificando en el mismo la información resumida sobre las asignaciones de memoria que este ha realizado. Implementado en una máquina virtual local, haciendo uso de módulos de kernel para una distribución Arch Linux con versión 5.4 de Kernel, escribiendo archivos con esta información en la carpeta /proc para posteriormente leerlos en Golang, donde serán consultados por el Frontend elaborado con Vite donde se muestra la información de forma gráfica.



## OBJETIVOS

- Conocer el kernel de Linux y los módulos que actúan sobre el directorio /proc.
- Conocer la planificación de procesos de Linux.
- Investigar la función del directorio /proc para obtener la información del sistema.
- Poner en práctica los conocimientos sobre el Kernel de Linux.
- Familiarizarse con la terminal de Linux y comandos de sistema y usuario.
- Aprender a crear, monitorizar y montar procesos del Kernel de Linux.

# ESPECIFICACIÓN TÉCNICA

## Requisitos de Hardware y software

- Soporte de kernel y CPU de 64 bits para virtualización.
- Al menos 4 GB de RAM.
- Soporte de virtualización KVM.
- QEMU versión 5.2 o posterior.
- GCC 10.2.1-6
- Distribución de Linux (preferiblemente Debian GNU/Linux 11)

## Tecnologías utilizadas

- Arch Linux release 2023.06.01
- Included Kernel: 5.4
- Vite 4.3.9
- Golang 1.13

## DEPENDENCIAS

Se realizaron las siguientes instalaciones

- **GCC (instala Make también)**

```
sudo apt install build-essential
sudo apt-get install manpages-dev
```

- **Make**

```
sudo apt install make
```

- **Linux-headers (La versión del kernel de la distribución)**

```
sudo apt-get install linux-headers-$(uname -r)
```

- **Java**

```
sudo apt install default-jdk
java -version
```

- **Golang**

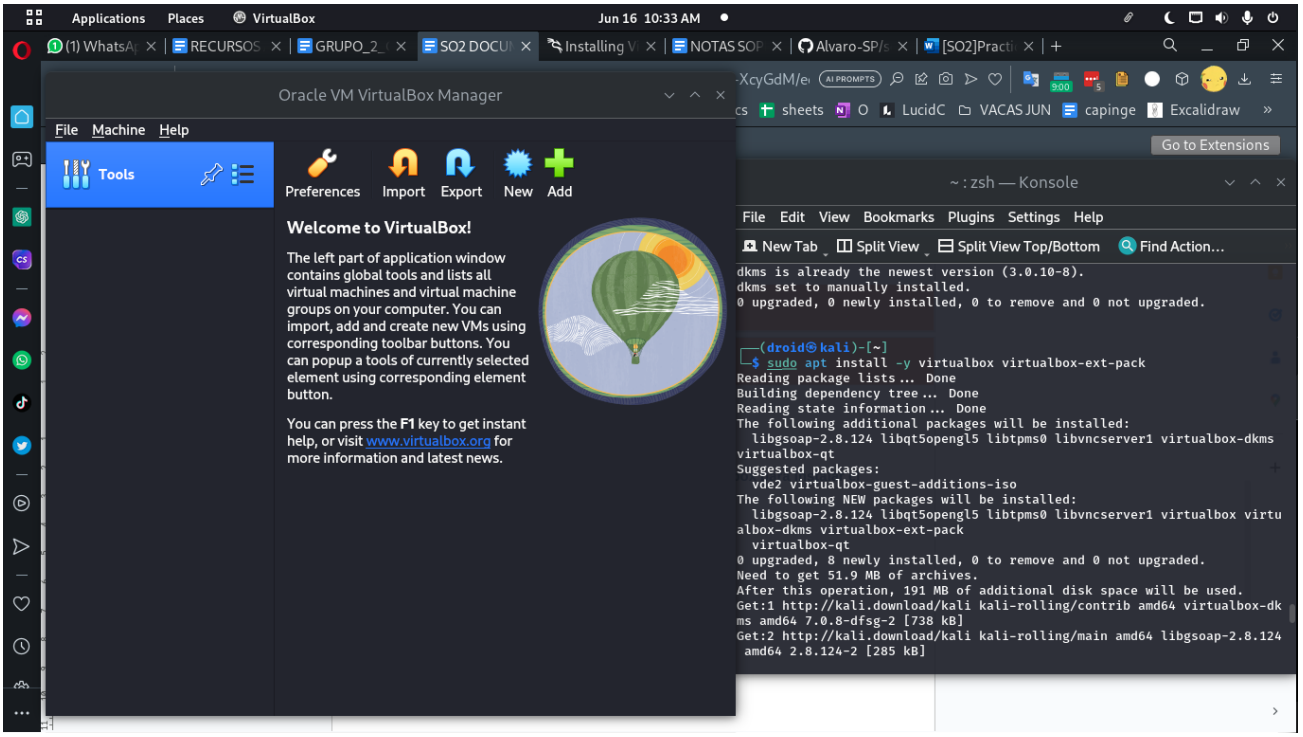
```
sudo apt install golang
go version
```

- **Git**

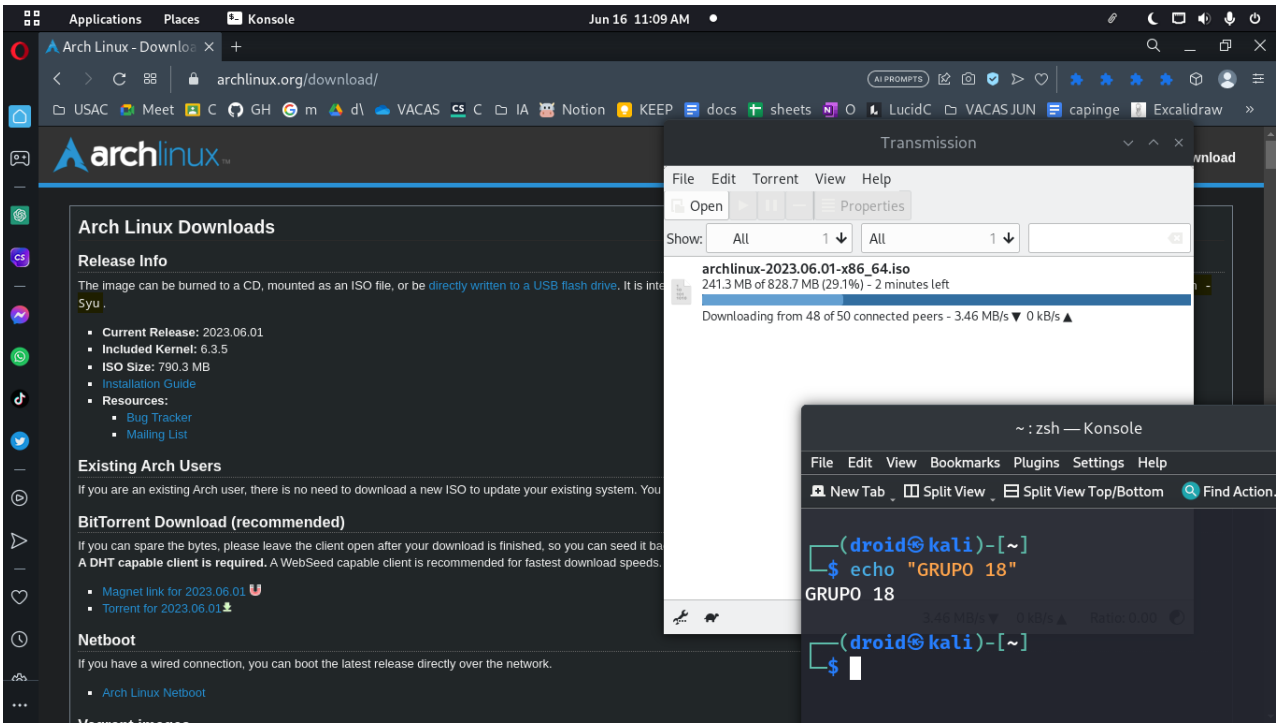
```
sudo apt install git
git -version
```

# INSTALACIÓN DE ARCH LINUX

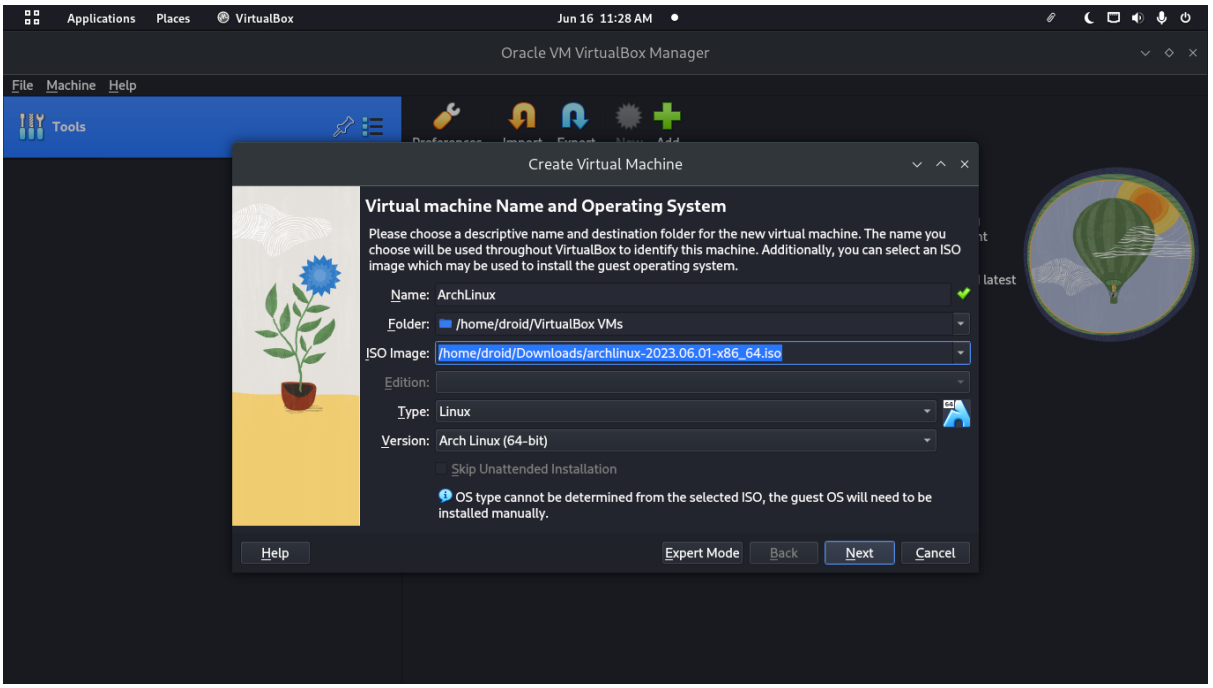
## Primero se ha descargado el software de Virtualbox para instalar el ISO

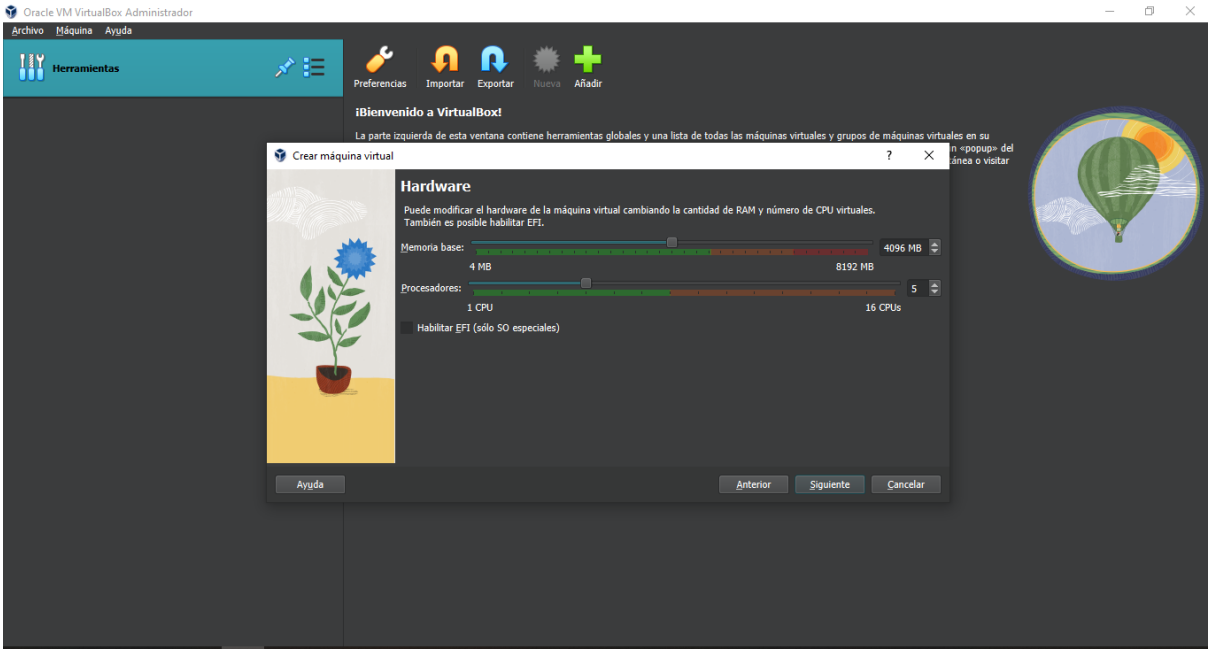


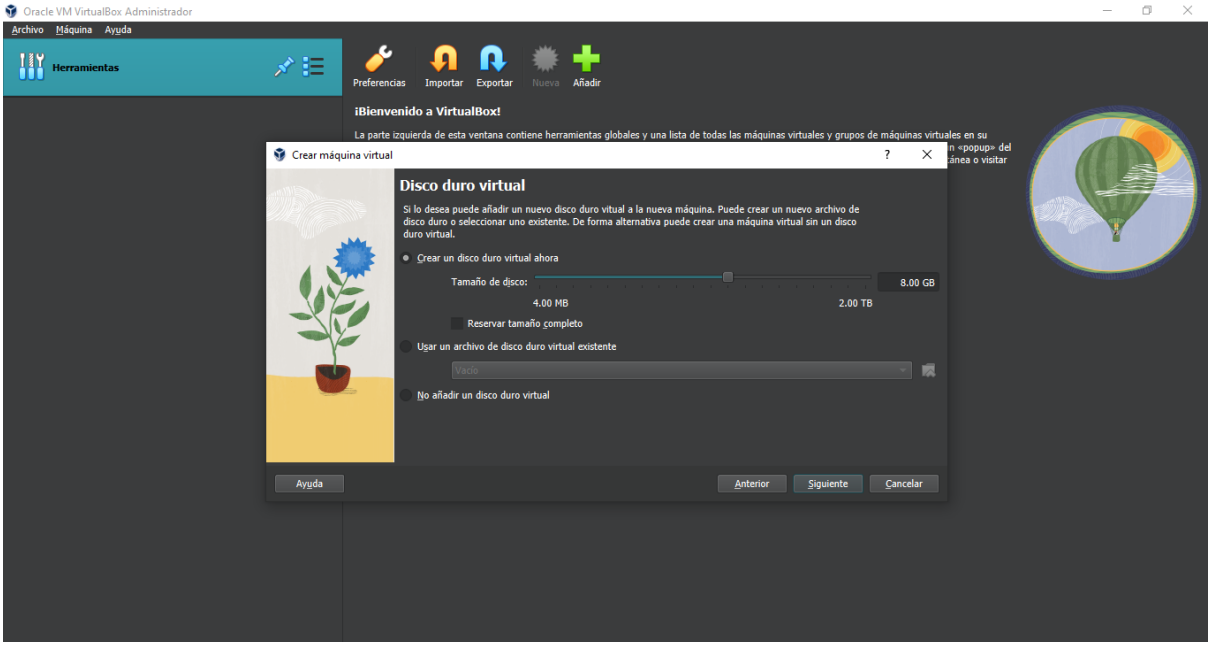

## Se ha descargado el iso



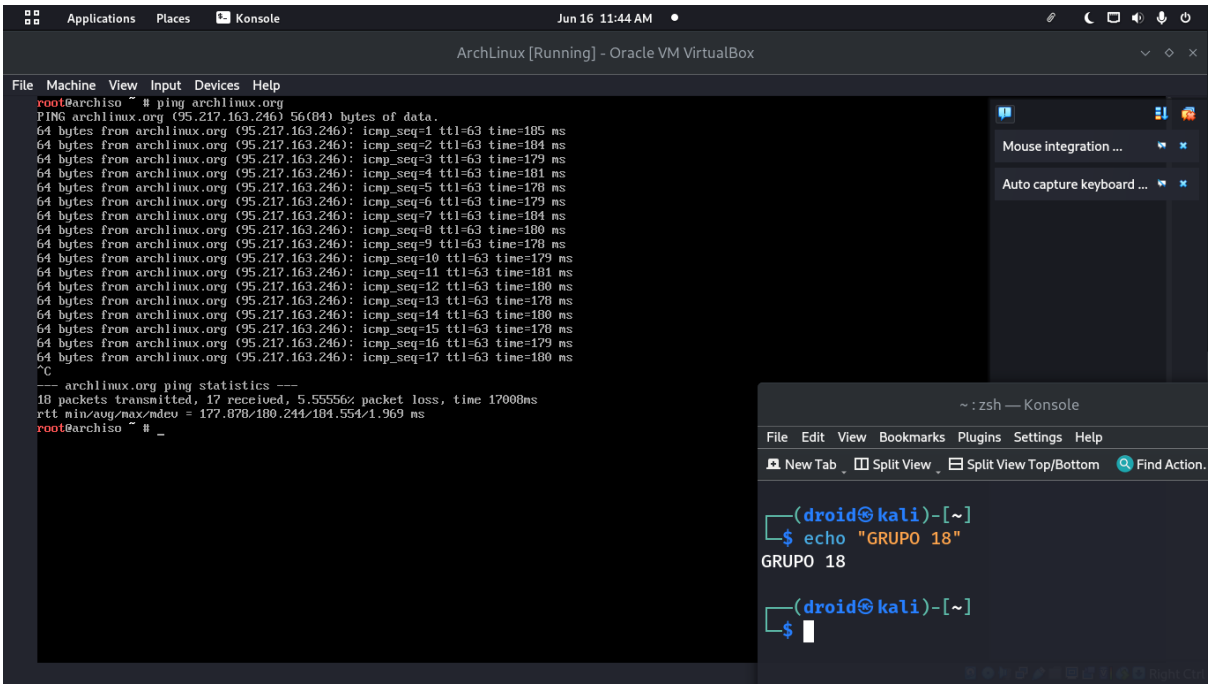
## Seleccionamos la imagen y sus recursos:



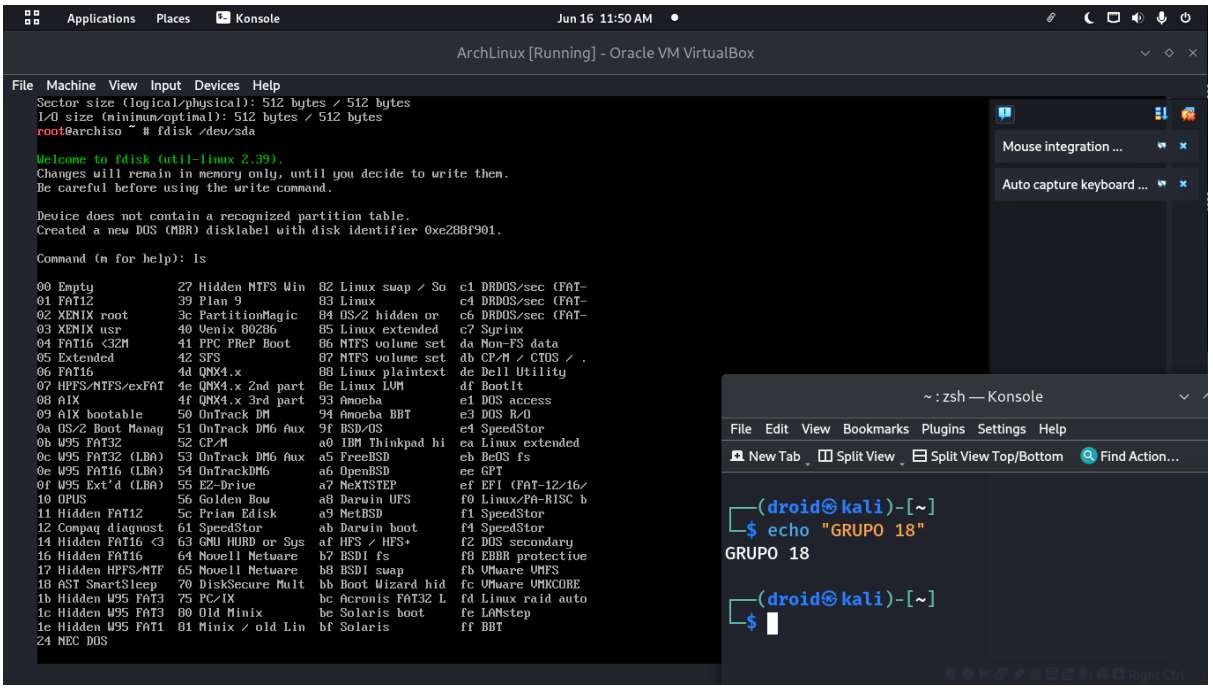



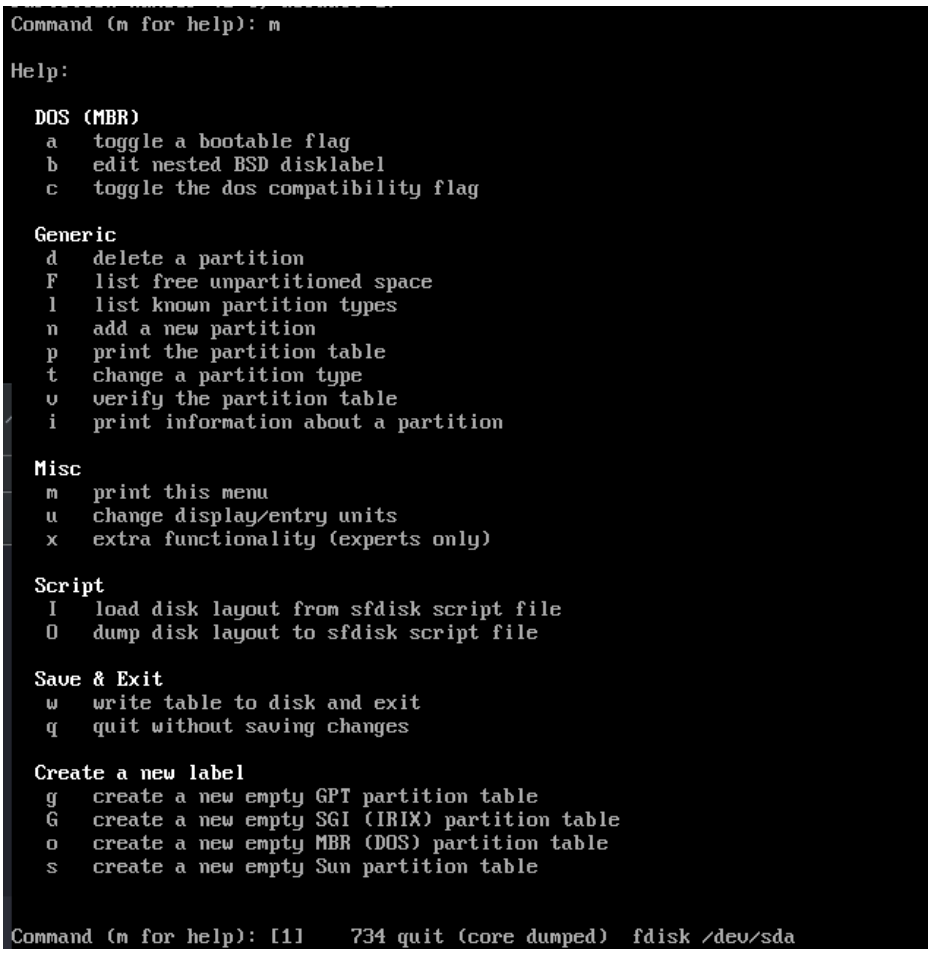



## Antes de seguir al iniciar la maquina verificamos conexión a internet

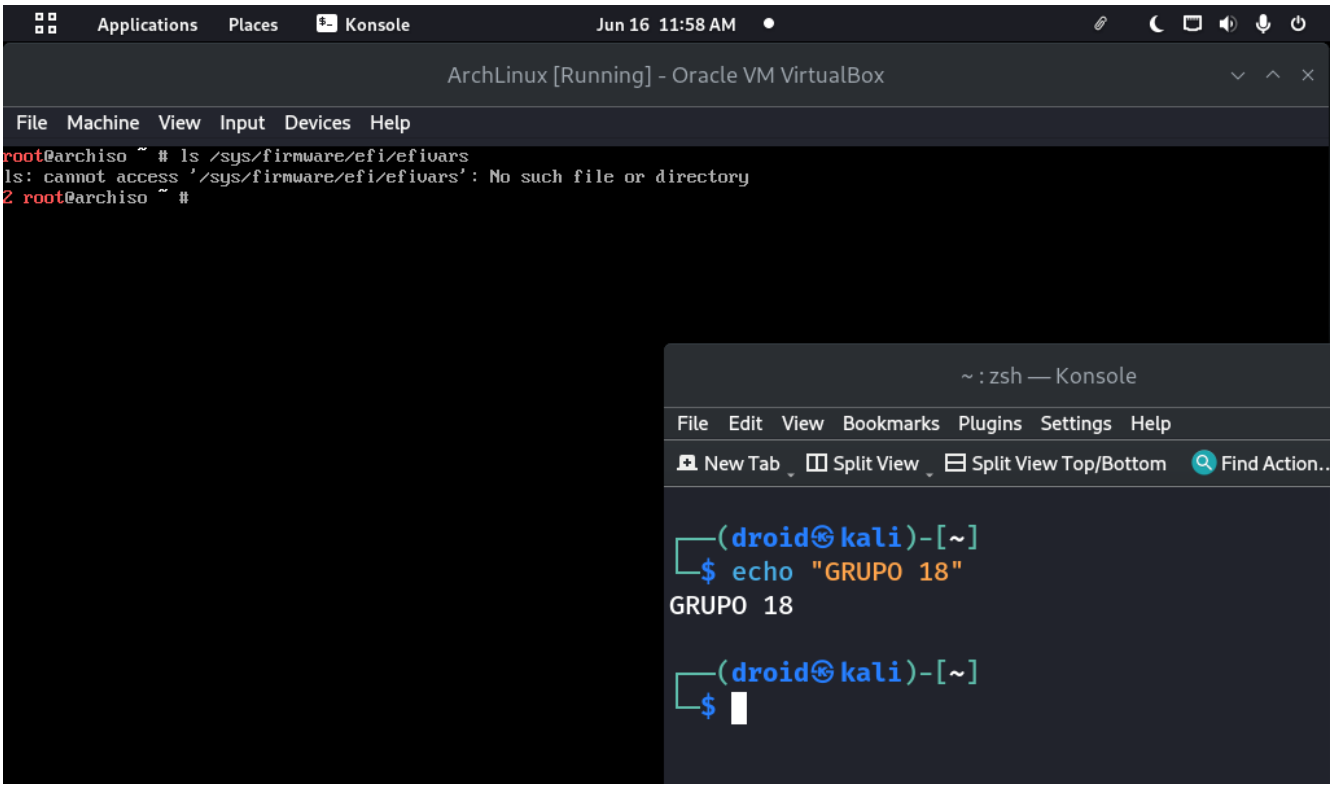


**Posteriormente particionamos el disco:**
**fdisk -l ⇐ sirve para ver los discos**
**fdisk /dev/sda ⇐ sirve para**

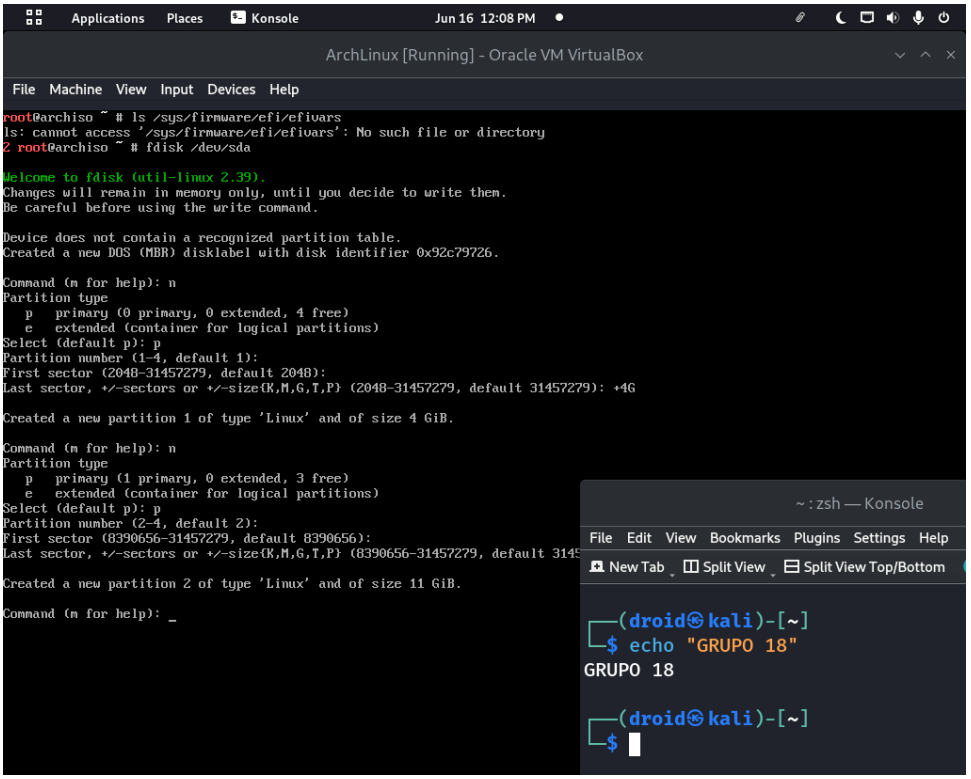


**Se verifica que no este en EFI (debe decir que no puede acceder):**

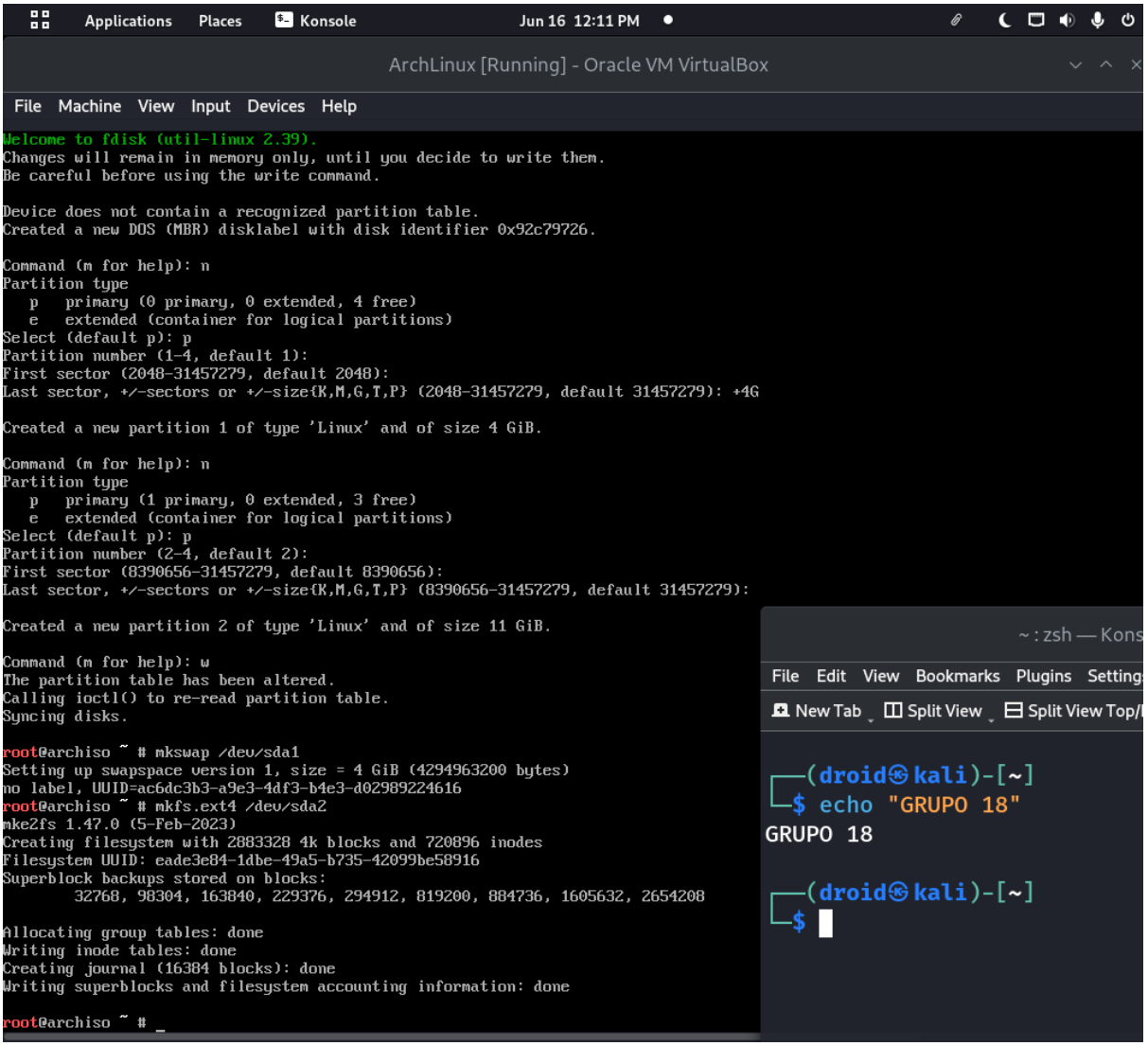



**Luego se crea la particion:**
**SWAP y ROOT**

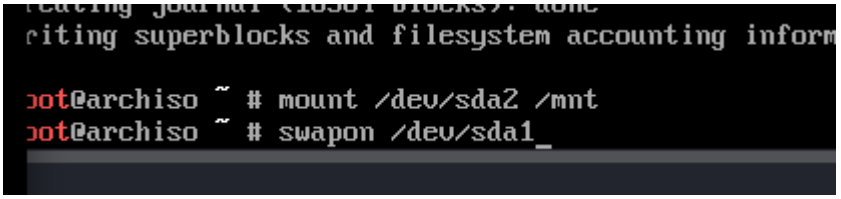


## Antes de seguir presionamos “w” que es para guardar tabla en disco y salir

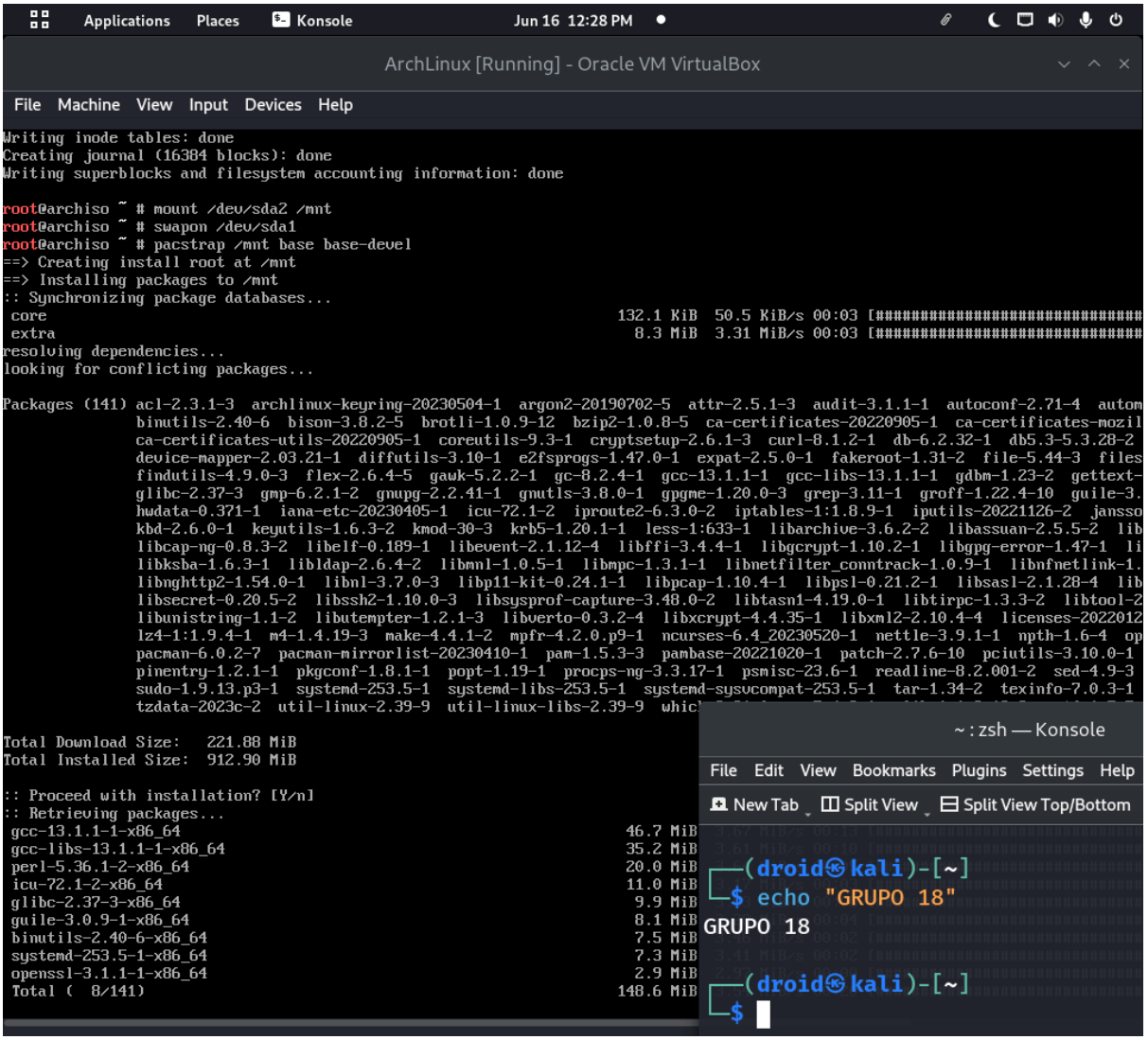

## Luego formateamos SWAP y la otra con ext4



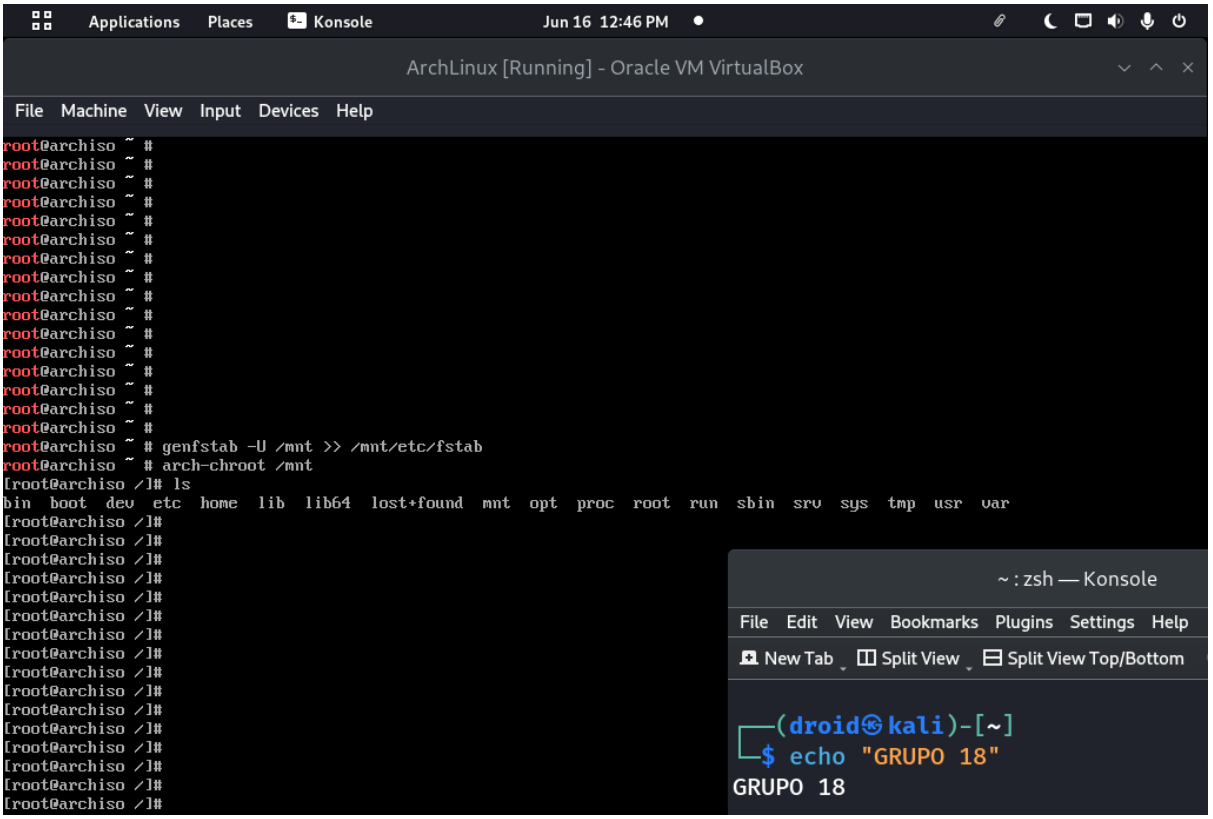
**Montamos la partición y el sistema de archivos con: mount /dev/sda2 /mnt y también: swapon /dev/sda1**




**Descargar los paquetes para la instalación de arch**
**pacstrap /mnt base base-devel**



**genfstab -U /mnt >> /mnt/etc/fstab**
**arch-chroot /mnt**



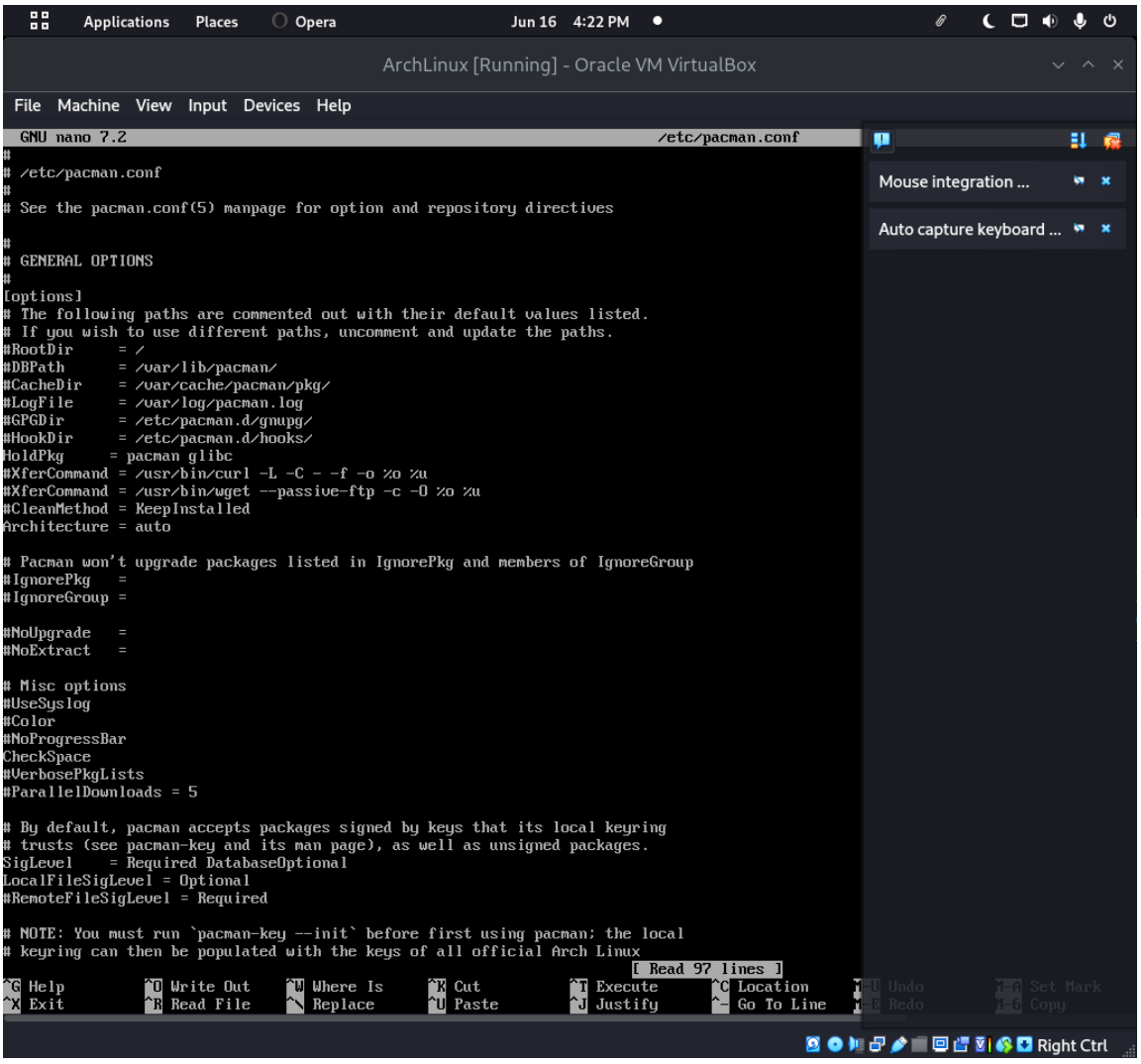

**Ahora cambiamos la zona horaria a la mas cercana que es la de Mexico**
**Hay varias regiones pero elegimos la de mexico:**
**ln -sf /usr/share/zoneinfo/Mexico/BajaSur /etc/localtime**

**Luego se instala Nano con el comando <<pacman -S nano>>**
**si surge problema con el <enter> se ejecuta stty sane:**

```
[root@archiso /]# stty sane
[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   207 KiB/s 00:03 [##############################]
(1/1) checking keys in keyring                                                  [##############################]
(1/1) checking package integrity                                                [##############################]
(1/1) loading package files                                                     [##############################]
(1/1) checking for file conflicts                                               [##############################]
(1/1) checking available disk space                                             [##############################]
:: Processing package changes...
(1/1) installing nano                                                           [##############################]
:: Running post-transaction hooks...
(1/2) Arming ConditionNeedsUpdate...
(2/2) Updating the info directory file...
[root@archiso /]# _
```

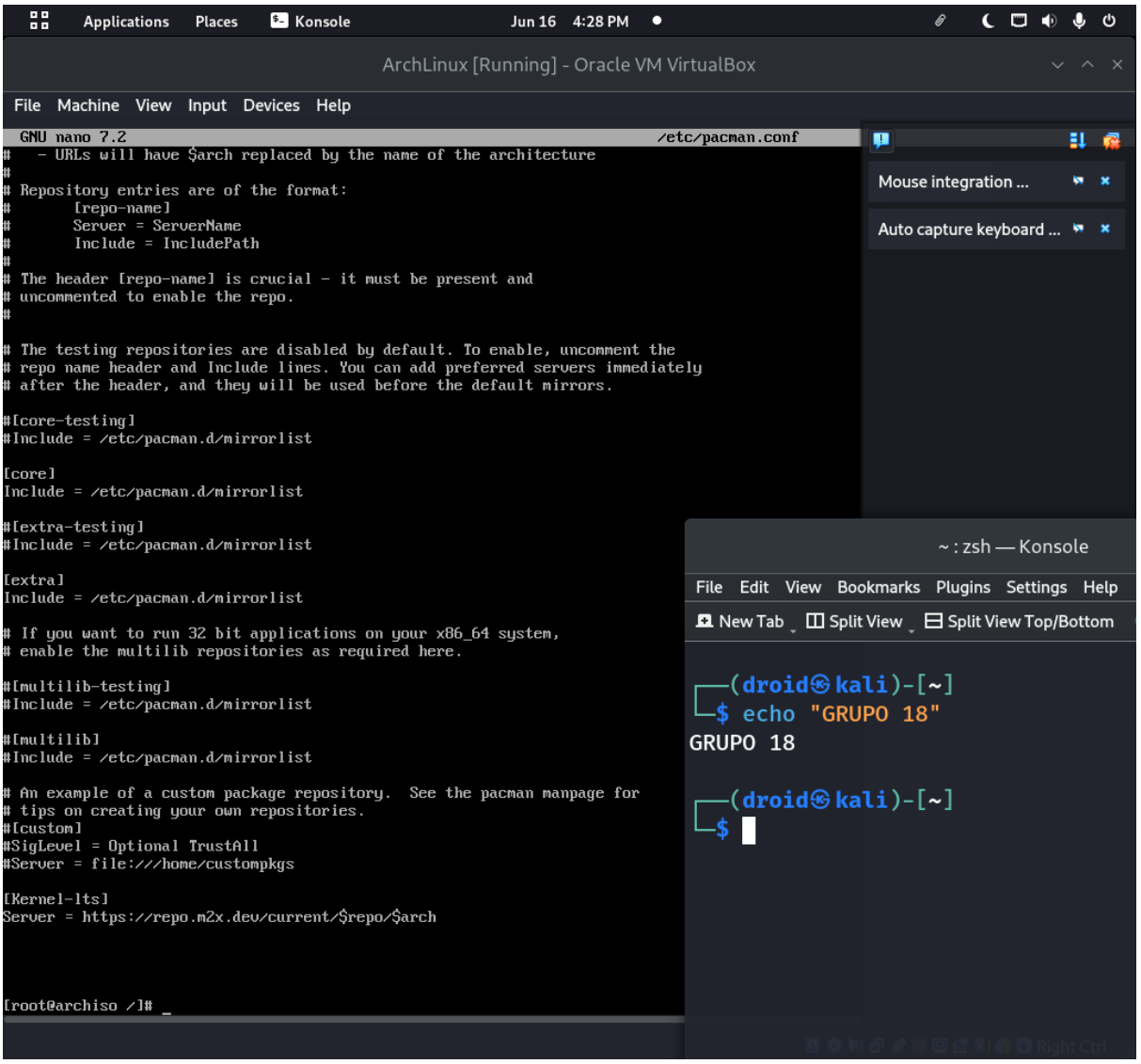
## INSTALACIÓN DEL KERNEL:

**Como no tenemos una versión anterior ejecutamos:  nano /etc/pacman.conf**



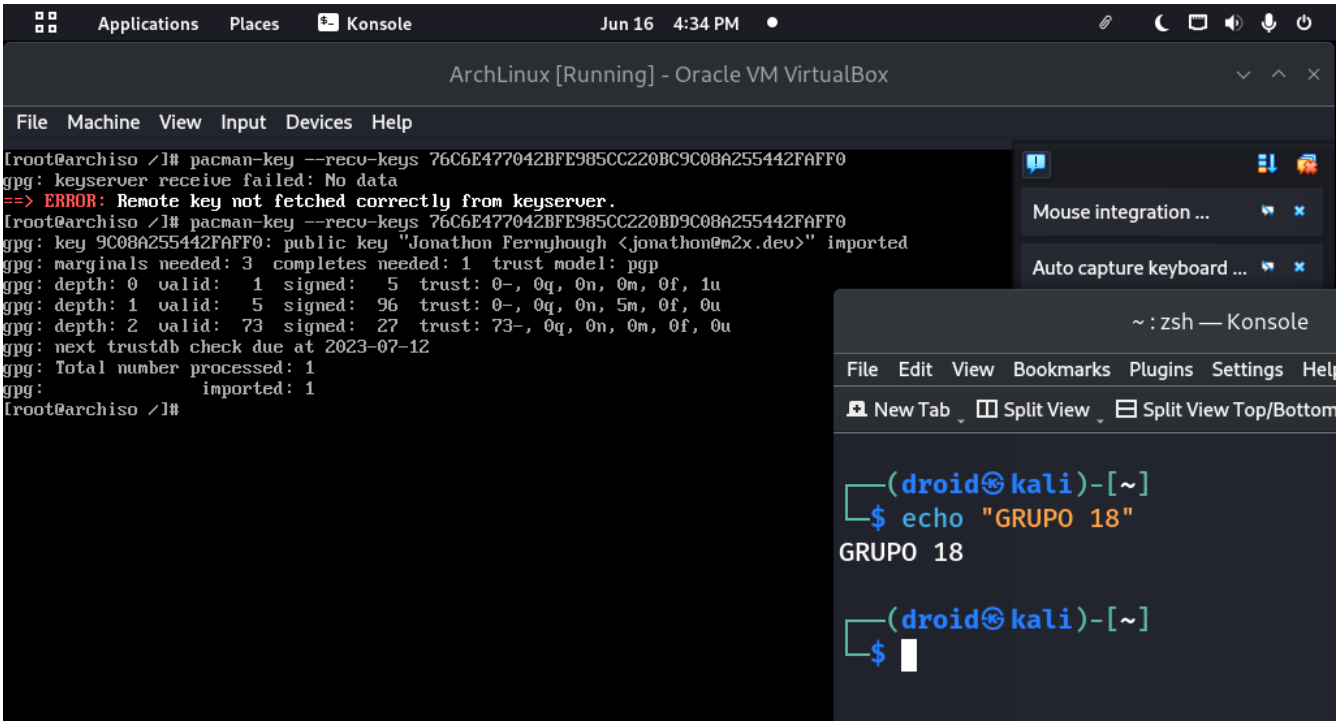

```
#
# /etc/pacman.conf
#
# See the pacman.conf(5) manpage for option and repository directives

#
# GENERAL OPTIONS
#
[options]
# The following paths are commented out with their default values listed.
# If you wish to use different paths, uncomment and update the paths.
#RootDir     = /
#DBPath      = /var/lib/pacman/
#CacheDir    = /var/cache/pacman/pkg/
#LogFile     = /var/log/pacman.log
#GPGDir      = /etc/pacman.d/gnupg/
#HookDir     = /etc/pacman.d/hooks/
HoldPkg     = pacman glibc
#XferCommand = /usr/bin/curl -L -C - -f -o %o %u
#XferCommand = /usr/bin/wget --passive-ftp -c -O %o %u
#CleanMethod = KeepInstalled
Architecture = auto

# Pacman won't upgrade packages listed in IgnorePkg and members of IgnoreGroup
#IgnorePkg   =
#IgnoreGroup =

#NoUpgrade   =
#NoExtract   =

# Misc options
#UseSyslog
#Color
#NoProgressBar
CheckSpace
#VerbosePkgLists
#ParallelDownloads = 5

# By default, pacman accepts packages signed by keys that its local keyring
# trusts (see pacman-key and its man page), as well as unsigned packages.
SigLevel    = Required DatabaseOptional
LocalFileSigLevel = Optional
#RemoteFileSigLevel = Required

# NOTE: You must run `pacman-key --init` before first using pacman; the local
# keyring can then be populated with the keys of all official Arch Linux
```

**Se agrega el repositorio dentro de ese archivo:**

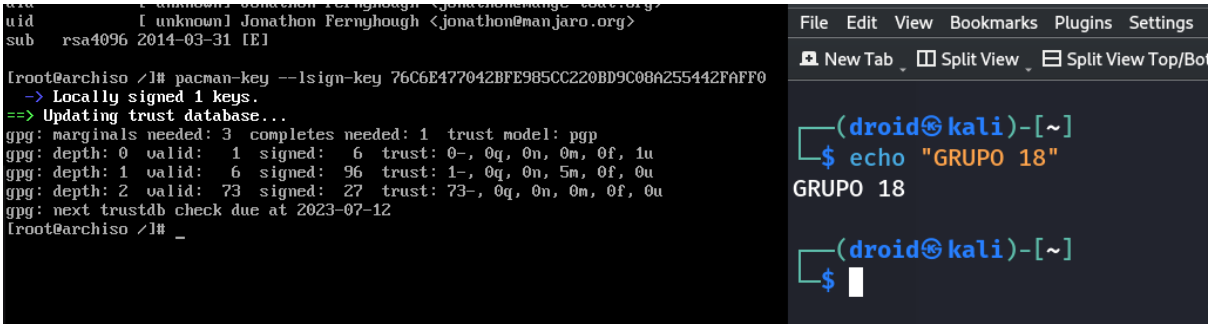


## AGREGAR LAS LLAVES

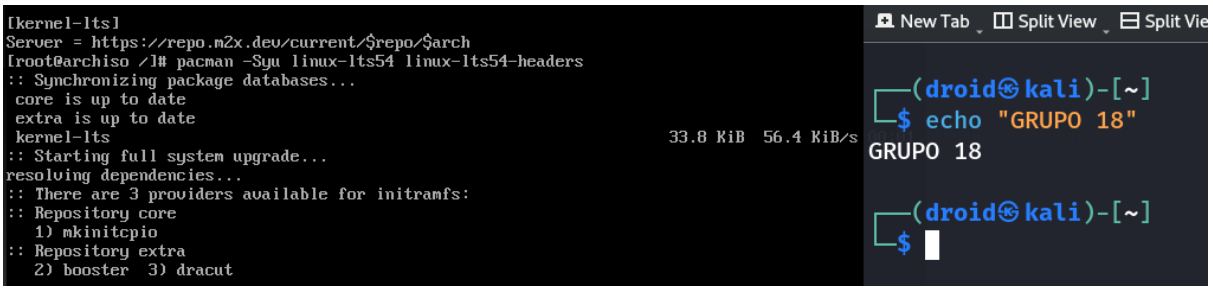
## pacman-key –recv-keys 76C6E477042BFE985CC220BC9C08A255442FAFF0



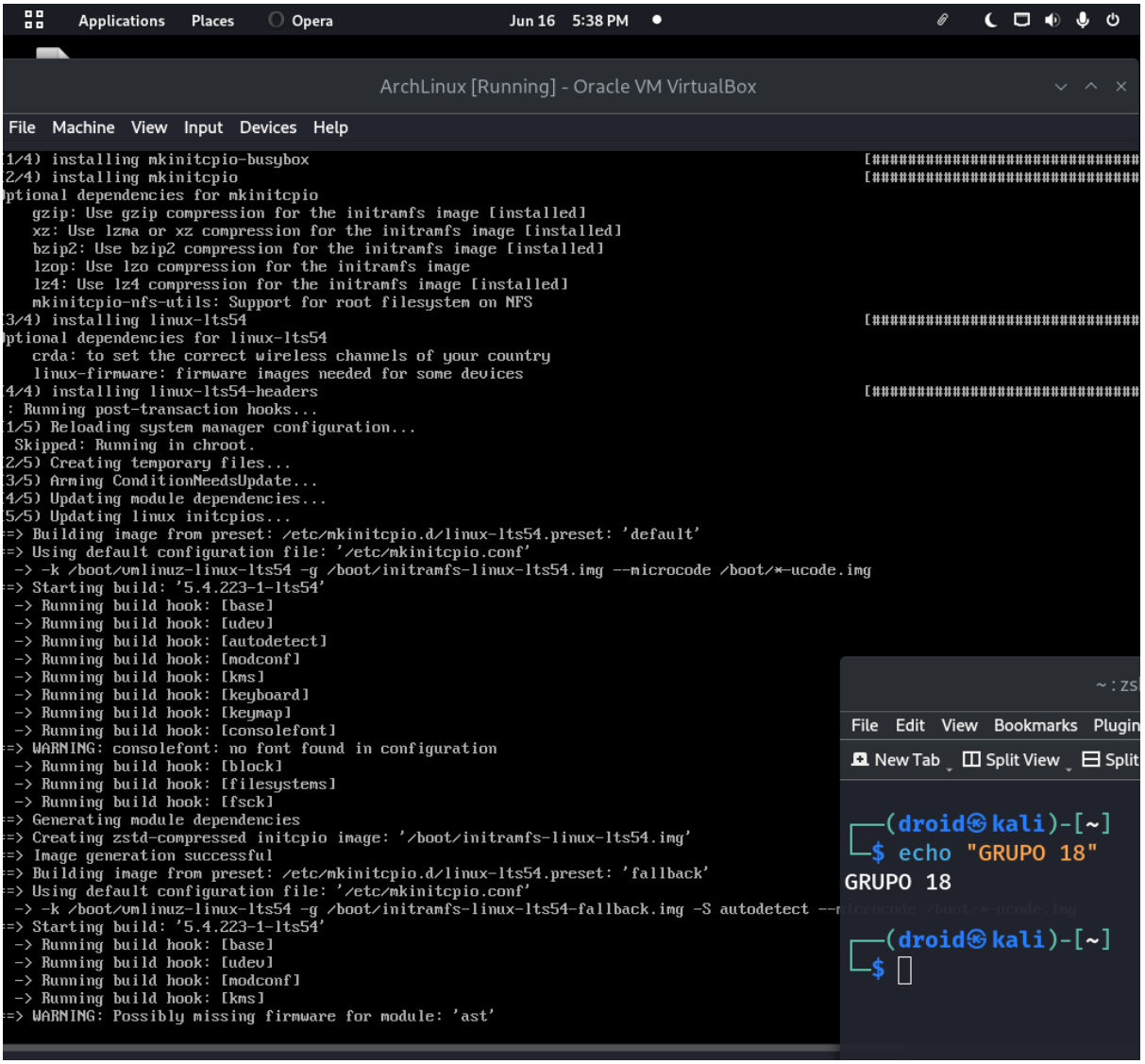

## AHORA firmamos con el flag –finger y con –lsign-key

## Se instala el siguiente repositorio: pacman -Syu linux-lts54 linux-lts54-headers

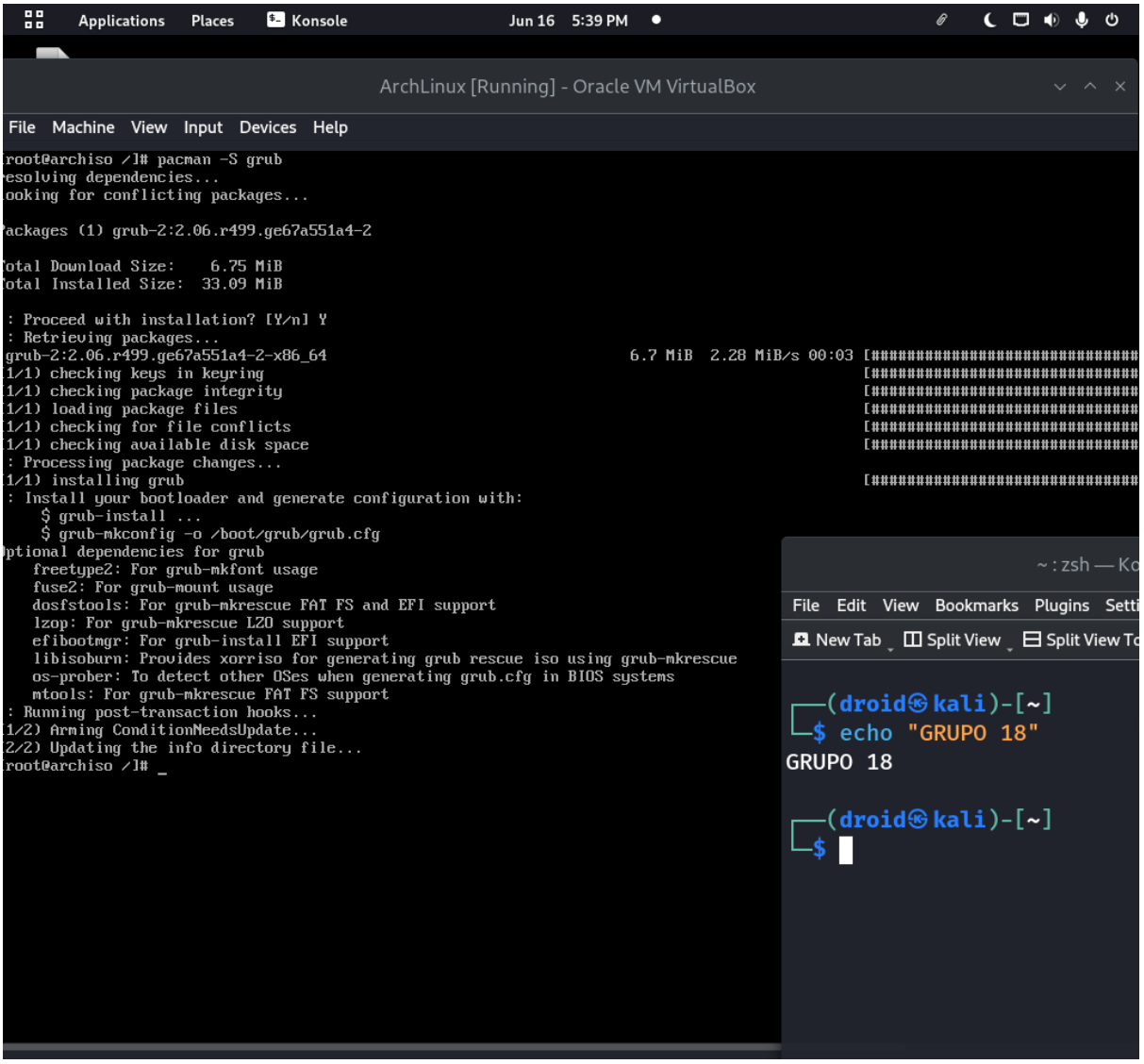



## Entonces dejamos todo por defecto y seguimos:

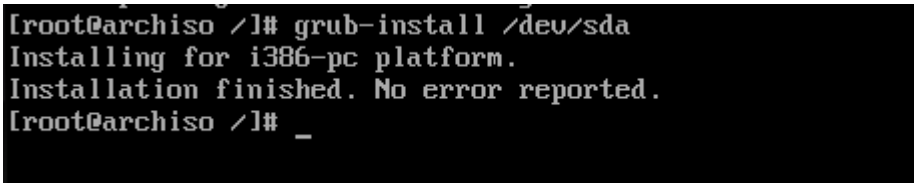


## INSTALACION DE GRUB

## pacman -S grub




## grub-install /dev/sda ← ponemos el disco entero



**grub-mkconfig -o /boot/grub/grub.cfg**

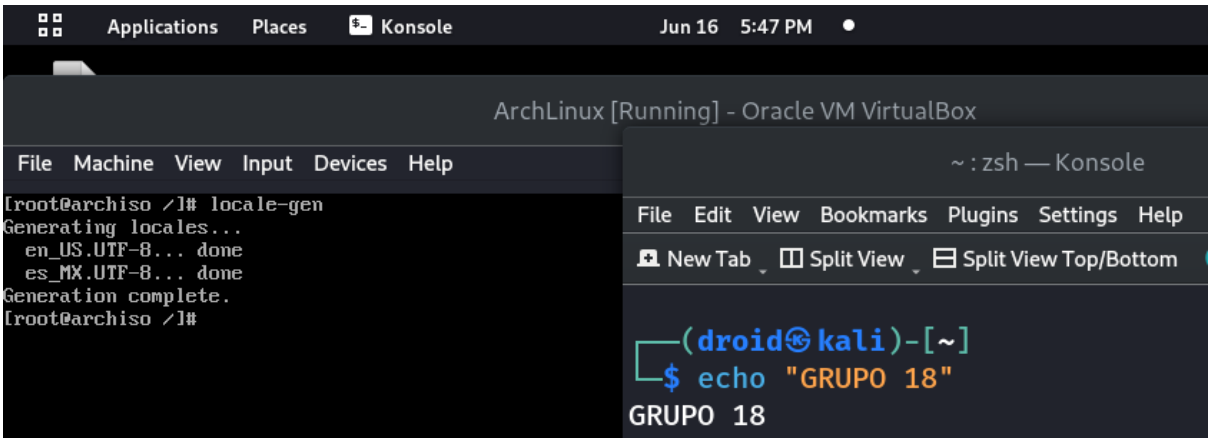
```
[root@archiso /]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux-lts54
Found initrd image: /boot/initramfs-linux-lts54.img
Found fallback initrd image(s) in /boot:  initramfs-linux-lts54-fallback.img
Warning: os-prober will not be 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
[root@archiso /]# _
```

**AGREGANDO LOS IDIOMAS:**

**nano /etc/locale.gen**

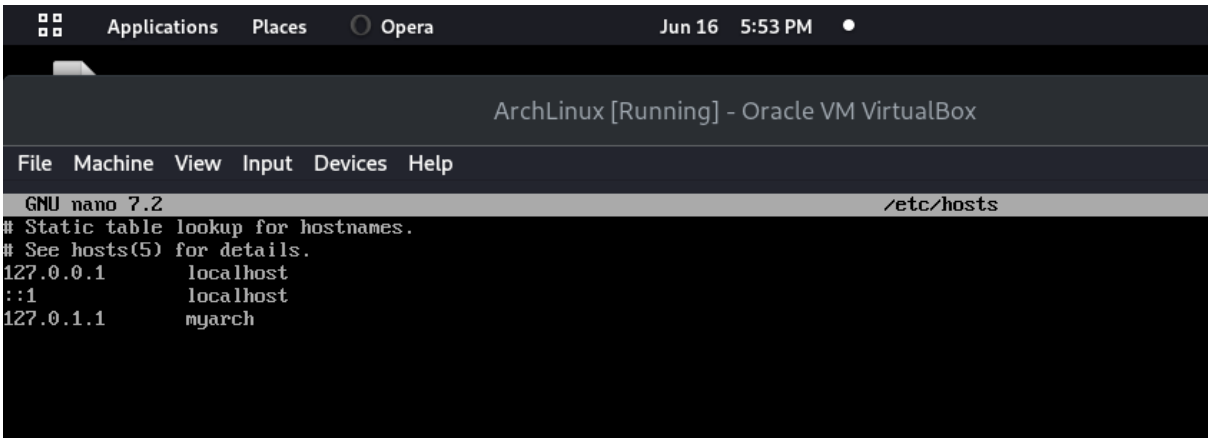
```
en_US.UTF-8 UTF-8
#en_US ISO-8859-1
```

```
es_MX.UTF-8 UTF-8
#es_MX ISO-8859-1
```

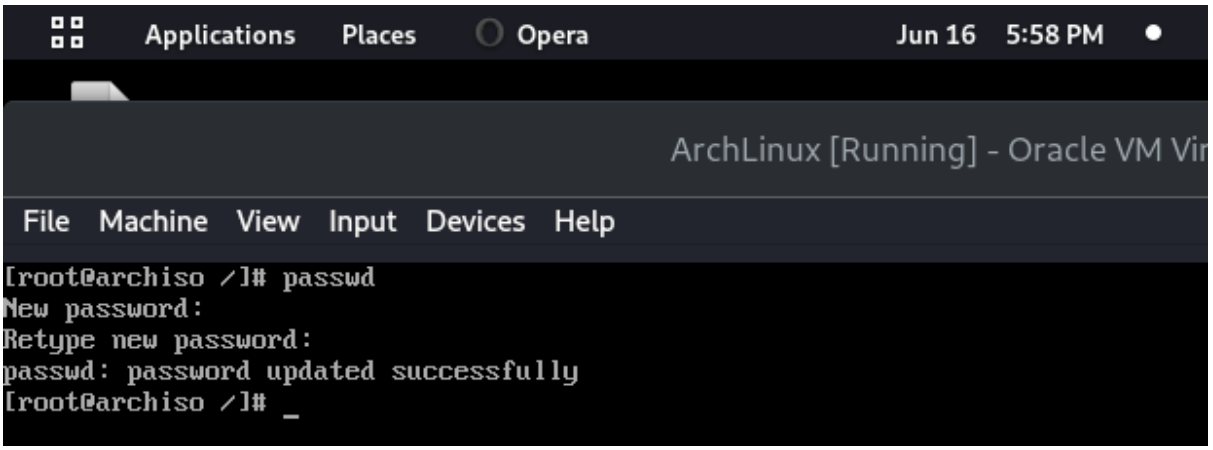



**AGREGO LA RED:**

```
[root@archiso /]# echo LANG=en_US.UTF-8 > /etc/locale.conf
[root@archiso /]# export LANG=en_US.UTF-8
[root@archiso /]# echo myarch > /etc/hostname
[root@archiso /]# touch /etc/hosts
[root@archiso /]# nano /etc/hosts
```



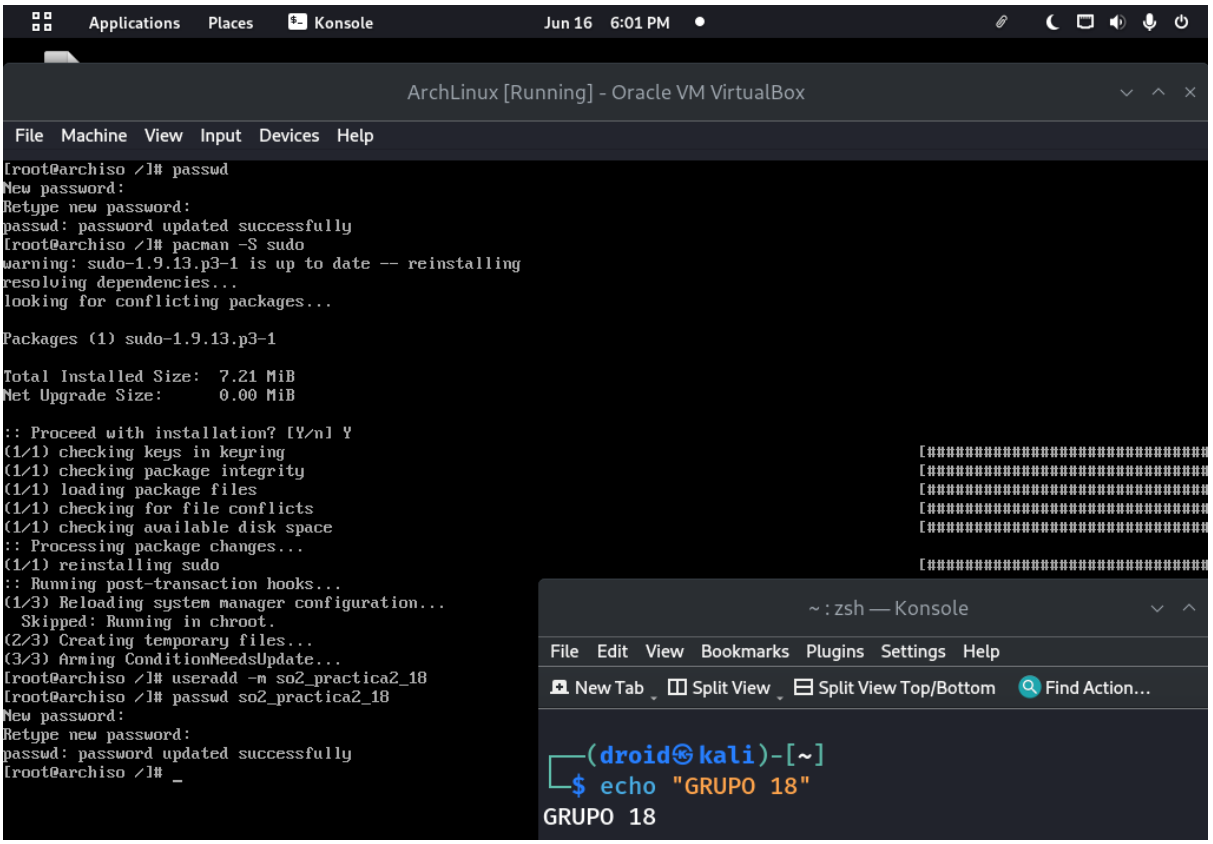
**passwd 123**




**CREACIÓN DE USUARIO:**

**pacman -S sudo**

**se agrega el usuario so2_practica2_18**

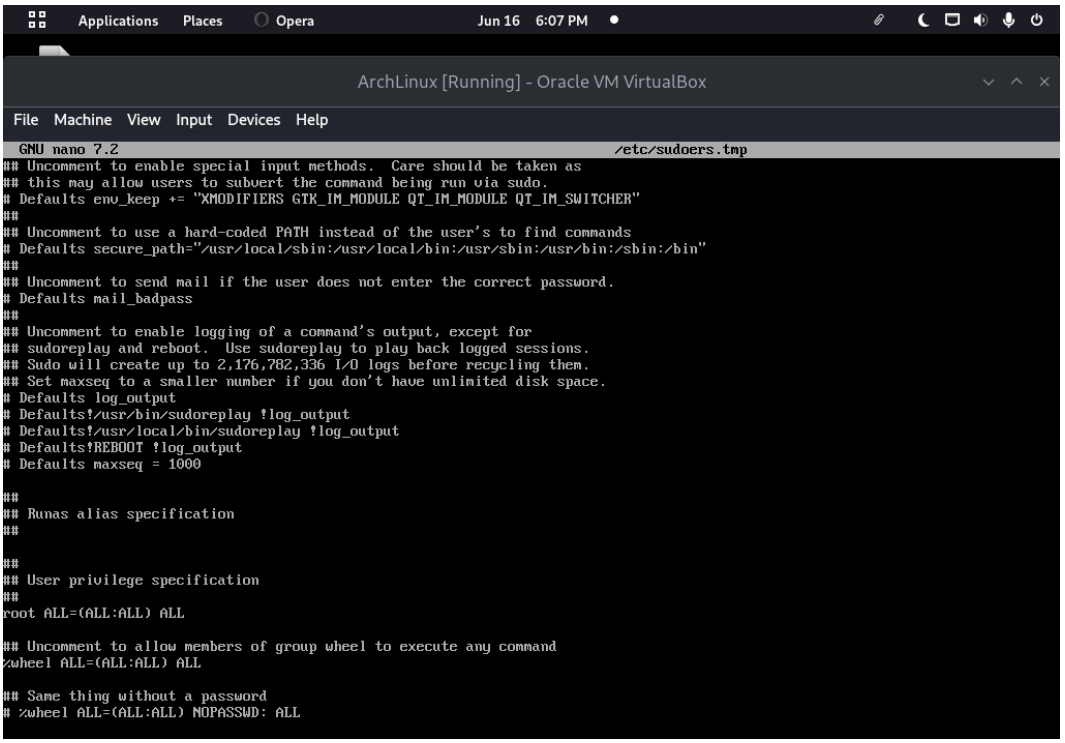


**damos permisos**

**usermod -aG wheel,audio,video,storage so2_practica2_18**

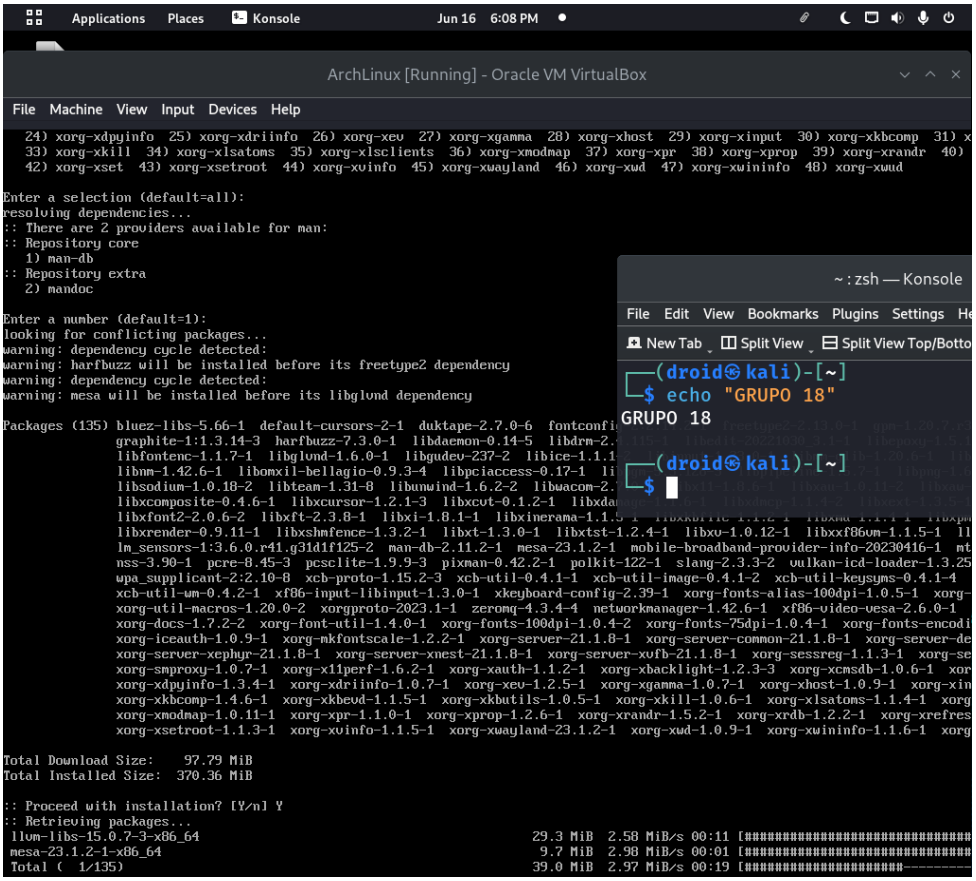
**EDITOR=nano visudo**

**luego en nano descomentamos la linea Uncomment to allow memnbers of group…**

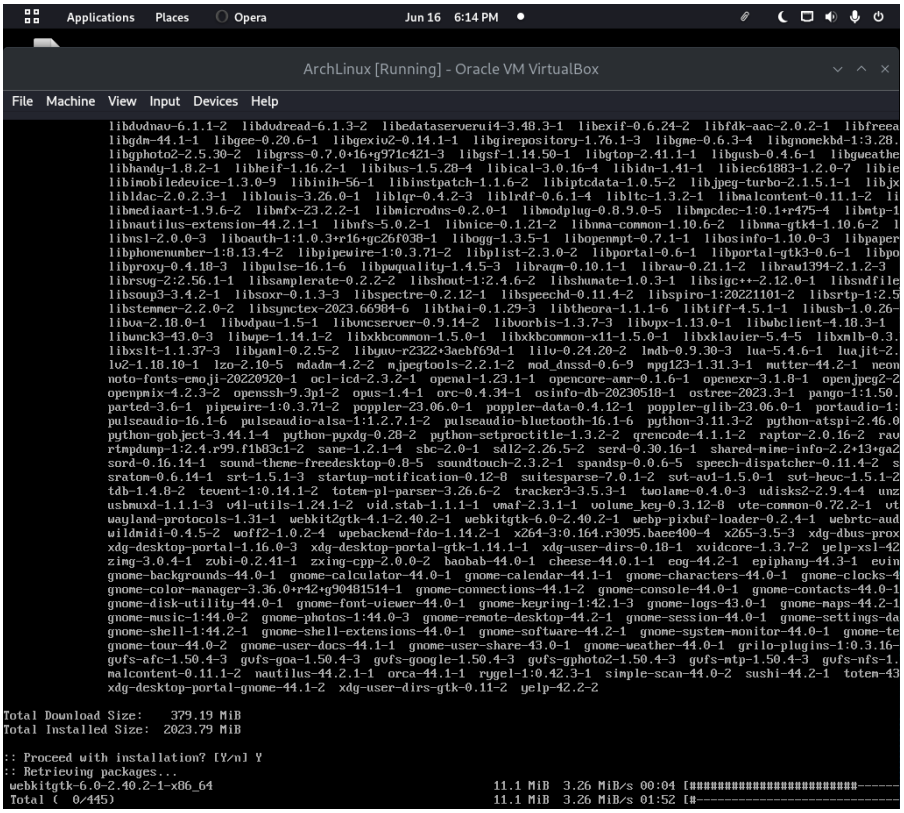


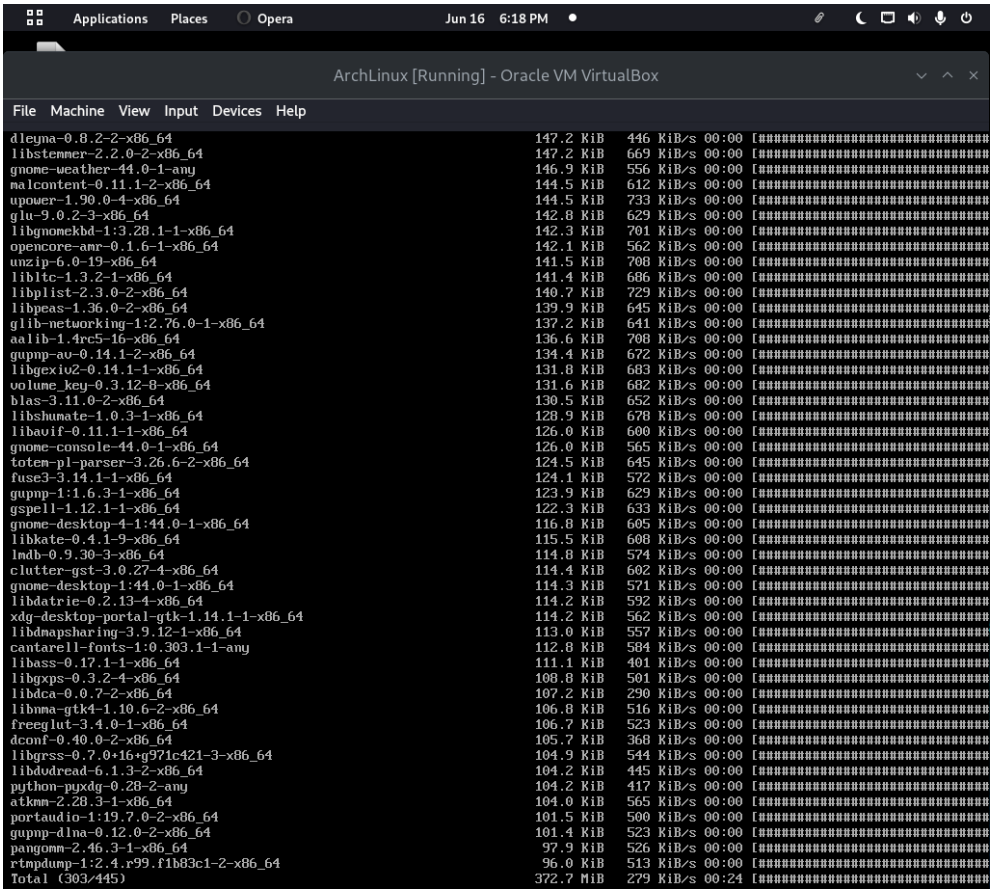
## INSTALACIÓN GRÁFICA:

### pacman -S xorg networkmanager

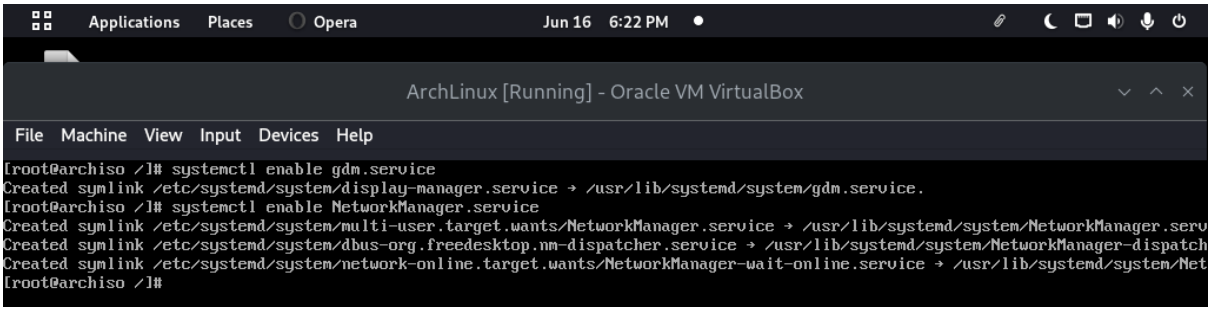



### luego el comando pacman -S gnome

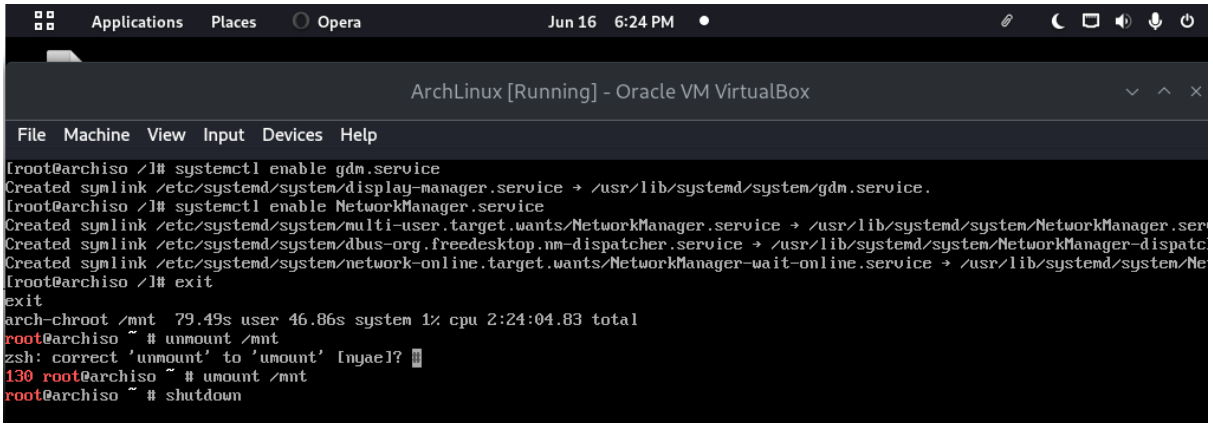




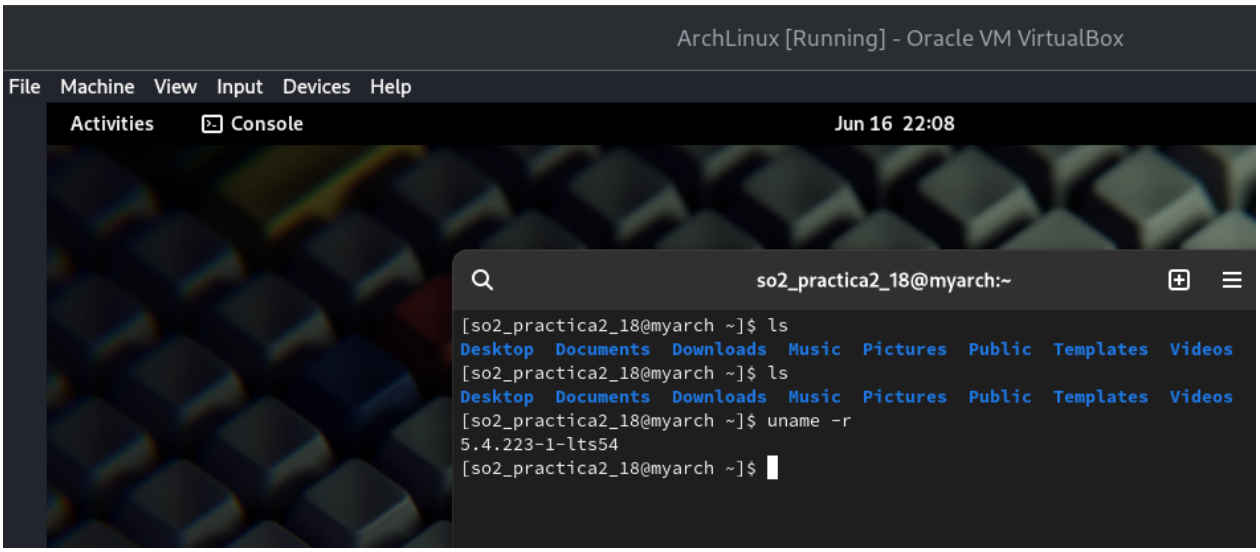

## Activar el network manager y el adaptador de arch

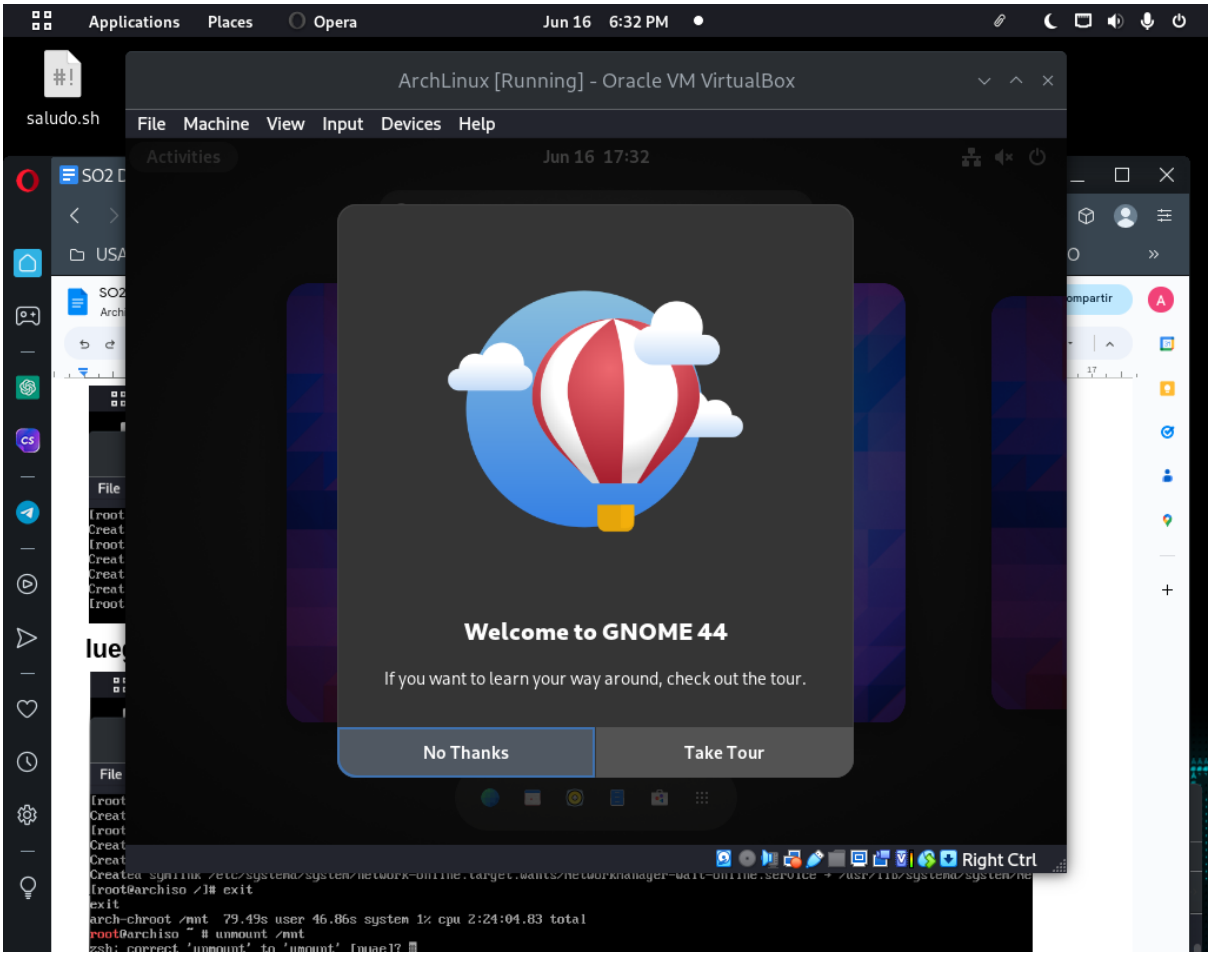


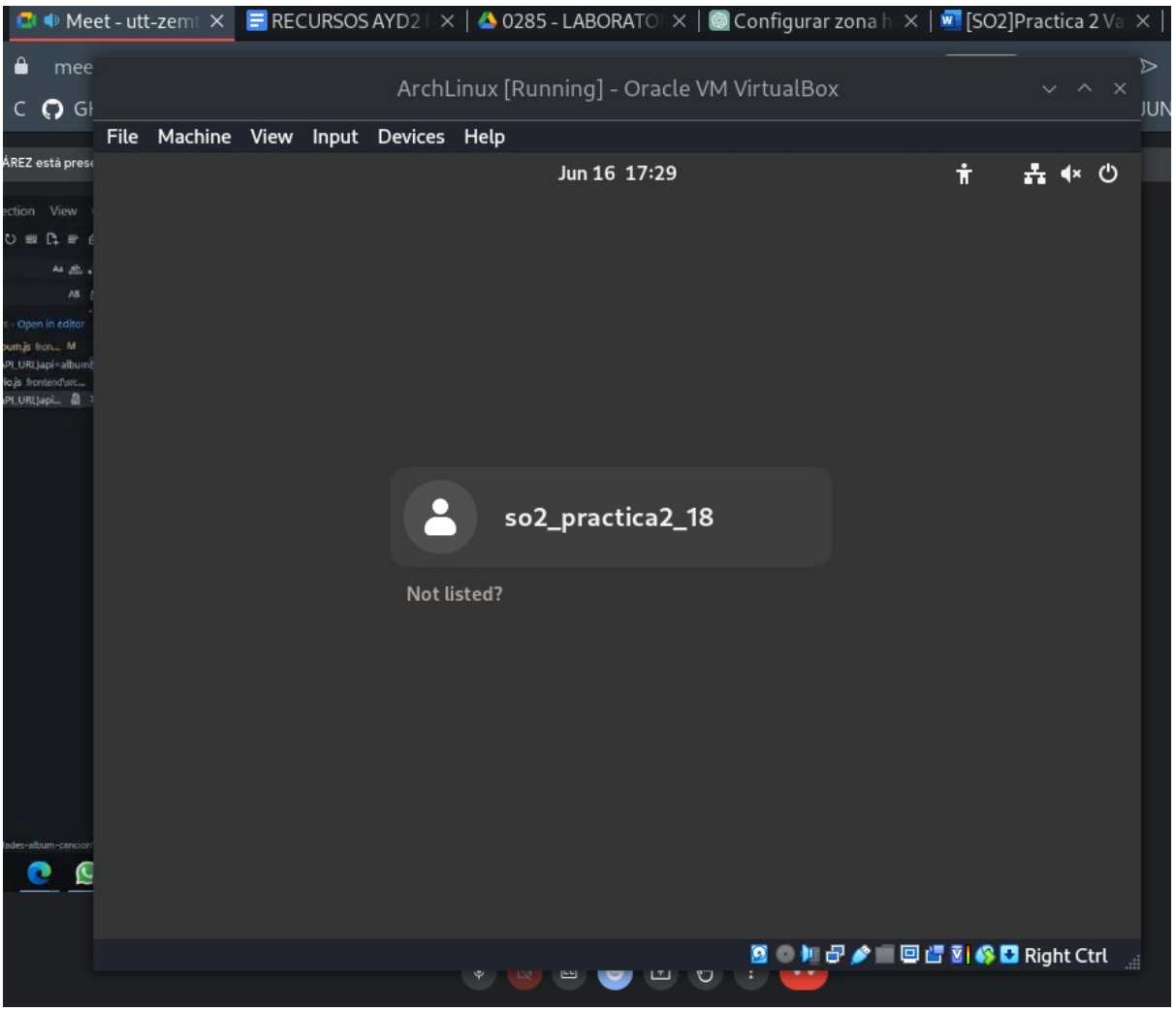

## luego desmontamos la instalación y listo



**shutdown now**




Meet - utt-zem
RECURSOS AYD2
0285 - LABORATO
Configurar zona h
[SO2]Practica 2 Va
ArchLinux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jun 16 17:29
so2_practica2_18
Not listed?
Right Ctrl

## LÓGICA DEL PROGRAMA (MÉTODOS PRINCIPALES)

### Módulos de Kernel

#### Módulo de memoria RAM

Para lograr obtener el porcentaje de memoria RAM utilizado por la máquina virtual se requirió leer "sysinfo", de donde se obtuvo la memoria total y la libre, para que más adelante sea calculado el porcentaje en el backend de Golang.

```
static void getMemoria(struct seq_file *archivo)
{
// Total de memoria
long memorytotal;
si_meminfo(&infsys);
memorytotal = infsys.totalram * infsys.mem_unit; // bytes
seq_printf(archivo, "\t\t\"MEMORIA_TOTAL\":%ld,\n", memorytotal);

long memorylibre;
memorylibre = infsys.freeram * 1000; // bytes
seq_printf(archivo, "\t\t\"MEMORIA_LIBRE\":%ld,\n", memorylibre);

long memorybuffer;
memorybuffer = infsys.bufferram * 1000; // bytes
seq_printf(archivo, "\t\t\"BUFFER\":%ld\n,", memorybuffer);

seq_printf(archivo, "\t\t\"MEM_UNIT\":%d\n,", infsys.mem_unit);

long porcentaje;
porcentaje = (((memorytotal) - (infsys.freeram * infsys.mem_unit) -
(infsys.bufferram * infsys.mem_unit) - (infsys.sharedram *
infsys.mem_unit)) * 10000) / (memorytotal);
seq_printf(archivo, "\t\t\"PORCENTAJE\":%ld\n", porcentaje);
}
```

**Módulo de CPU para procesos del sistema**

Para lograr obtener los procesos y sus procesos hijos que son utilizados por el sistema se utilizó "task_struct", obteniendo por cada proceso su PID, nombre, usuario, estado y porcentaje de ram utilizado.

```
static int escribir_archivo(struct seq_file *archivo, void *v)
{
struct task_struct *task;
struct task_struct *task_hijo;
struct list_head *children;
long memproc;
long memproc2;
int indext = 0; // indice para el nombre de proceso
struct file *file;
struct file *file2;
char *strstate = ""; // variable para guardar el estado del
proceso
char buffer[256];
int len;
long cpu_usage = 0;
struct sysinfo info;
long mem_usage;
bool first = true; // solo para el primer proceso la coma
long memoria_total = 0;
// variables para guardar cantidad de procesos
long int ejecucion = 0;
long int suspendido = 0;
long int detenido = 0;
long int zombie = 0;
long int totales = 0;

//!     ------------------------------     CALCULO     DEL     CPU
------------------------------
// ? https://www.anshulpatel.in/posts/linux_cpu_percentage/
long total_time_prev = 0;
long used_time_prev = 0;

for_each_process(task)
{
total_time_prev += get_total_time(task);
used_time_prev += task->utime + task->stime;
}
```

```
// Sleep for 1 second
msleep(500);

long total_time = 0;
long used_time = 0;

// Traverse the task list to calculate total and used CPU time
for_each_process(task)
{
total_time += get_total_time(task);
used_time += task->utime + task->stime;
}

long total_time_diff; //= total_time - total_time_prev;
long used_time_diff;
// Calculate the CPU percentage
if (total_time > total_time_prev)
{
if (total_time < total_time_prev)
{
total_time_diff = total_time_prev - total_time;
}
else
{
total_time_diff = total_time - total_time_prev;
}
if (used_time < used_time_prev)
{
used_time_diff = used_time_prev - used_time;
}
else
{
used_time_diff = used_time - used_time_prev;
}

cpu_usage = (used_time_diff * 100) / total_time_diff;
}
printk(KERN_INFO "cpu_usage: %ld%%\n, total_time: %ld%%\n
total_time_prev: %ld%%\n used_time: %ld%%\n used_time_prev:
%ld%%\n", cpu_usage, total_time, total_time_prev, used_time,
used_time_prev);
printk(KERN_INFO "total_time_diff: %ld%%\n", total_time_diff);
```

```
printk(KERN_INFO "used_time_diff: %ld%%\n", used_time_diff);
// printk(KERN_INFO "CPU Percent: %d%%\n", cpu_usage);

si_meminfo(&info);

// total_mem = (info.totalram * info.mem_unit) >> 10; // ! memoria
total en MB
// printk(KERN_INFO "Total memory: %lu mB\n", total_mem/1000);
memoria_total = (info.totalram * info.mem_unit) >> 10;
//     printk(KERN_INFO     "Total     memory:     %lu     MB\n",
(memoria_total/1000000));
seq_printf(archivo, "{\n");
seq_printf(archivo, "\"cpu_usage\":"); //* "cpu_usage": 25.35,
seq_printf(archivo, "%ld , \n", cpu_usage);
seq_printf(archivo,     "\"data\":     [");     //*     "data":     {
"proceso1":{"pid":   254,   ...   ,   "procesoshijos":   [...]"},
"proceso2":{...}, ... },
for_each_process(task)
{
if (!first)
{
seq_printf(archivo, ",");
}
//! 0 : ejecutando
//! 4 : zombie
//! 8 : detenido
//! 1 o 1026 : suspendido
if (task->mm)
{
memproc = (get_mm_rss(task->mm) << (PAGE_SHIFT - 10));
// printk(KERN_INFO "Memoria de %s: %lu MB", task->comm, memproc);
mem_usage  =  ((memproc  *  100)  /  (memoria_total  >>  10));  //!
PORCENTAJE CON 2 DECIMALES PARSEAR EN FRONT
//  printk(KERN_INFO  "Porcentaje  de  memoria  de  %s:  %lu  %%\n",
task->comm,mem_usage);
}
if (task->state == 0 || task->state == 1026 || task->state == 2)
{
ejecucion++;
strstate = "ejecucion";
}
else if (task->state == 4)
{
```

```
zombie++;
strstate = "zombie";
}
else if (task->state == 8 || task->state == 8193)
{
detenido++;
strstate = "detenido";
}
else if (task->state == 1 || task->state == 1026)
{
suspendido++;
strstate = "suspendido";
}
totales++;
/* Get the passwd structure for the UID */
// char *nombre_usuario = get_cred_username(task->real_cred);

seq_printf(archivo, "{\"id\": \"%d_%s\",\"pid\": %d, \"nombre\":
\"%s\", \"usuario\": \"%d\", \"estado\": \"%s\", \"ram\": %lu,
\n\"procesoshijos\": [",
indext,
task->comm,
task->pid,
task->comm,
task->cred->uid,
strstate, mem_usage);
indext++;
task_lock(task);
children = &(task->children);
list_for_each_entry(task_hijo, children, sibling)
{
if (task_hijo->mm)
{
// memproc2 = (get_mm_rss(task_hijo->mm)<<PAGE_SHIFT)/(1024*1024);
// ! memoria de cada proceso hijo
// mem_usage = (memproc2*10000 / (long)(memoria_total/1000000));

memproc = (get_mm_rss(task_hijo->mm) << (PAGE_SHIFT - 10));
// printk(KERN_INFO "Memoria de %s: %lu MB", task->comm, memproc);
mem_usage = ((memproc * 100) / (memoria_total >> 10));
}
/* Get the passwd structure for the UID */
// pw = getpwuid(task_hijo->cred->uid.val);
```

```
seq_printf(archivo,    "{\"pid\":    %d,    \"nombre\":    \"%s\",
\"usuario\": \"%d\", \"estado\": \"%s\", \"ram\": %lu}",
task_hijo->pid,
task_hijo->comm,
task_hijo->real_cred->uid,
strstate,
mem_usage);

if (task_hijo->sibling.next != &task->children)
{
seq_printf(archivo, ",");
}
}
task_unlock(task);
seq_printf(archivo, "]\n}");
first = false;
}

seq_printf(archivo, "], \n");
seq_printf(archivo, "\"ejecucion\":");
seq_printf(archivo, "%li , \n", ejecucion);

seq_printf(archivo, "\"zombie\":");
seq_printf(archivo, "%li , \n", zombie);

seq_printf(archivo, "\"detenido\":");
seq_printf(archivo, "%li , \n", detenido);

seq_printf(archivo, "\"suspendido\":");
seq_printf(archivo, "%li , \n", suspendido);

seq_printf(archivo, "\"totales\":");
seq_printf(archivo, "%li \n", totales);
seq_printf(archivo, "}");

return 0;
}
```

# Backend de Golang

## Obtención de la memoria RAM

Con el siguiente código se busca el archivo creado en /proc por el módulo de kernel al obtener los datos de la memoria RAM

```
func RequestMemory() http.HandlerFunc {
return func(rw http.ResponseWriter, r *http.Request) {
salida, _, verificar := CMD("cat /proc/mem_grupo18")

if verificar != nil {
log.Printf("error: %v\n", verificar)
} else {
var dataJson Models.DATAJSONMEMORY
json.Unmarshal(salida.Bytes(), &dataJson)
rw.Header().Set("Content-Type", "application/json")
json.NewEncoder(rw).Encode(dataJson)
}
}
}
```

## Obtención de procesos del sistema

Con el siguiente código se busca el archivo creado en /proc por el módulo de kernel al obtener los datos de los procesos del sistema

```
func RequestPrincipal() http.HandlerFunc {
return func(rw http.ResponseWriter, r *http.Request) {

rw.Header().Set("Content-Type", "application/json")

salida, _, verificar := CMD("cat /proc/cpu_grupo18")
if verificar != nil {
log.Printf("error: %v\n", verificar)
return
}

var dataJson Models.CPUDATAJSON
if err := json.Unmarshal(salida.Bytes(), &dataJson); err != nil {

log.Printf("error: %v\n", err)
return
}

for i := range dataJson.DATA {

username, err := getUsername(dataJson.DATA[i].USUARIO)
```

```
if err != nil {
log.Printf("Error al obtener el nombre de usuario: %v\n", err)
http.Error(rw,       "Error       interno       del       servidor",
http.StatusInternalServerError)
return
}

dataJson.DATA[i].USUARIO = username

for j := range dataJson.DATA[i].PROCESOSHIJOS {
username,                              err                              :=
getUsername(dataJson.DATA[i].PROCESOSHIJOS[j].USUARIO)
if err != nil {
log.Printf("Error al obtener el nombre de usuario: %v\n", err)
http.Error(rw,       "Error       interno       del       servidor",
http.StatusInternalServerError)
return
}

dataJson.DATA[i].PROCESOSHIJOS[j].USUARIO = username


}
}
if err := json.NewEncoder(rw).Encode(dataJson); err != nil {
log.Printf("error: %v\n", err)
return
}
}
}
```

**Eliminar procesos**

El siguiente código se ejecuta cuando desde la interfaz el usuario desea eliminar uno de los procesos que se están ejecutando, utilizando el comando “sudo kill -9 {PID_proceso}”

```
func RequestKill() http.HandlerFunc {
return func(rw http.ResponseWriter, r *http.Request) {

if r.URL.Path != "/Kill" {
http.NotFound(rw, r)
return
}

if r.Method =="GET" {

id := r.URL.Query().Get("pid")
id = strings.TrimSuffix(id, "/")

_, _, verificar := CMD("sudo kill -9 " + id)

if verificar != nil {
log.Printf("error: %v\n", verificar)
} else {
fmt.Println("Eliminando Proceso: " + id)
}
}else{

rw.WriteHeader(http.StatusNotImplemented)
rw.Write([]byte(http.StatusText(http.StatusNotImplemented)))
}
}
}
```

## Maps de los procesos

Las siguientes funciones sirven para leer el archivo de maps del proceso obtenido en la petición post, cuando el usuario solicita la información detallada de cierto proceso. Siendo esta funcionalidad la que se implementó para cumplir con los nuevos requerimientos de la práctica.

```
func RequestMaps() http.HandlerFunc {
return func(rw http.ResponseWriter, r *http.Request) {

if r.URL.Path != "/maps" {
http.NotFound(rw, r)
return
}
if r.Method =="GET" {

id := r.URL.Query().Get("pid")
id = strings.TrimSuffix(id, "/")
num, err := strconv.Atoi(id)
if err != nil {
fmt.Println("Error al convertir el string a int:", err)
return
}

salida, verificar := ObtenerDatosMaps(num)
if verificar != nil {
log.Printf("error: %v\n", verificar)
return
}

rw.Header().Set("Content-Type", "application/json")
json.NewEncoder(rw).Encode(salida)
}else{

rw.WriteHeader(http.StatusNotImplemented)
rw.Write([]byte(http.StatusText(http.StatusNotImplemented)))
}
}
}
```

```
func ObtenerDatosMaps(pid int) ([]Models.MemoryMap, error) {

mapsPath := fmt.Sprintf("/proc/%d/maps", pid)
file, err := os.Open(mapsPath)
```

```
if err != nil {
return nil, err
}
defer file.Close()

var memoryMaps []Models.MemoryMap

reader := bufio.NewReader(file)

for {
line, err := reader.ReadString('\n')
if err != nil && err != io.EOF {
return nil, err
}

if line == "" || err == io.EOF {
break
}

fields := strings.Fields(line)
if len(fields) >= 6 {
addressRange := fields[0]
permissions := fields[1]
device := fields[3]
filePath := fields[5]


rangeFields := strings.Split(addressRange, "-")
if len(rangeFields) != 2 {
continue
}

start, err := strconv.ParseUint(rangeFields[0], 16, 64)
if err != nil {
log.Printf("Error al parsear la dirección de inicio: %v", err)
continue
}

end, err := strconv.ParseUint(rangeFields[1], 16, 64)
if err != nil {
log.Printf("Error al parsear la dirección de fin: %v", err)
continue
}
```

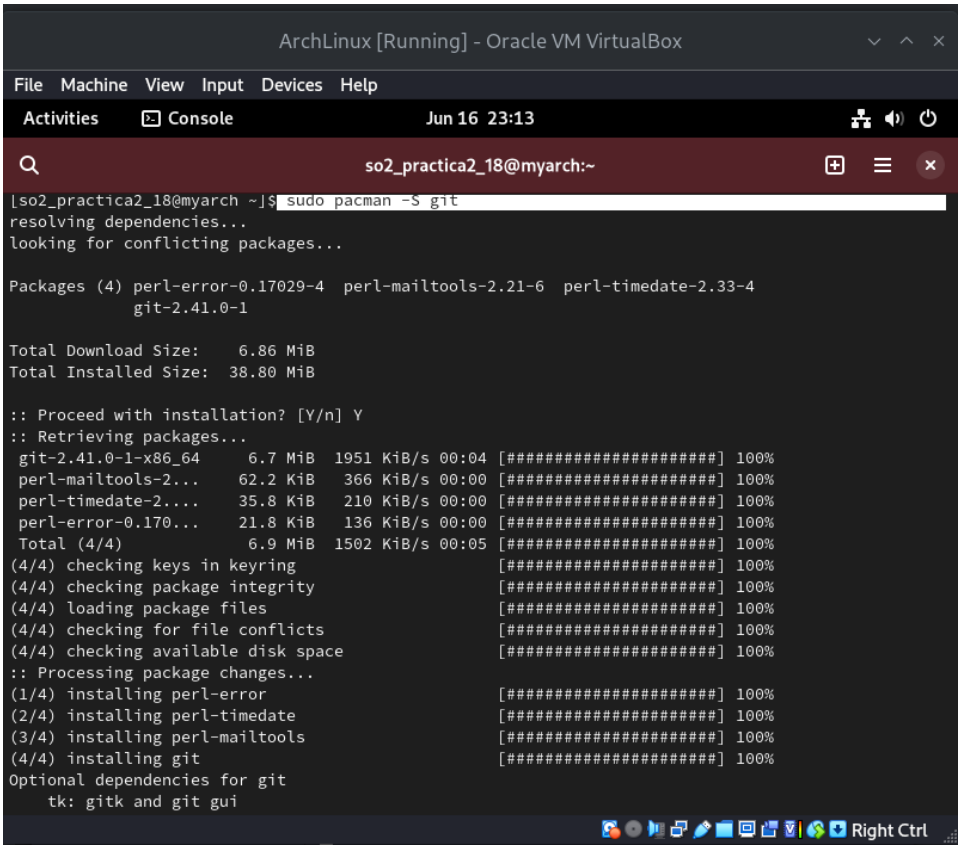
```
size := end - start

memoryMap := Models.MemoryMap{
Direccion: addressRange,
Tamanio: size,
Permisos: permissions,
Dispositivo: device,
Archivo: filePath,
}

memoryMaps = append(memoryMaps, memoryMap)
}
}

return memoryMaps, nil
}
```

En lo anterior cabe destacar el cálculo de la memoria que ocupa el proceso que se realiza restando el último espacio en memoria con el primero que ocupa el proceso

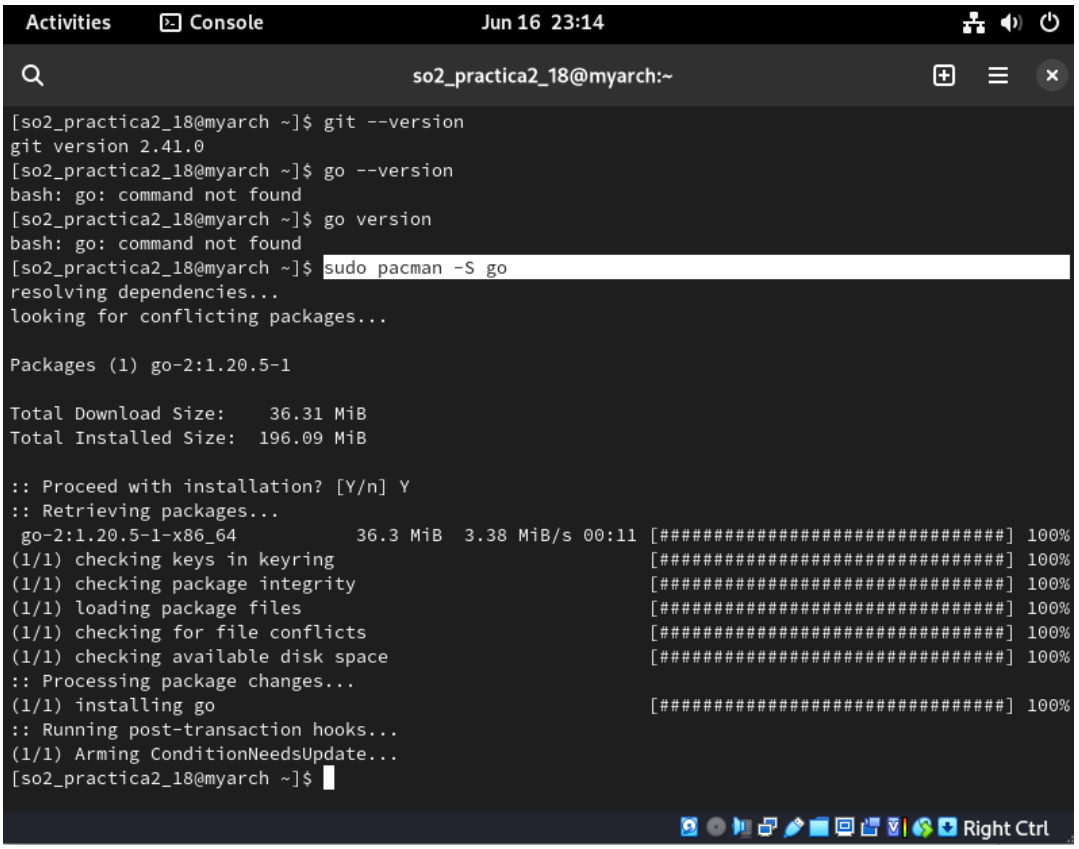
```
// Cálculo del tamaño de segmento
size := end - start
```

## INSTALACIÓN DE PAQUETES

### GIT




**GOLANG**




**sudo pacman -S linux-headers**

**Se modificó el código del módulo del cpu y de la memoria ram para que funcionara con la versión antigua del kernel, ya que la que utiliza arch linux es la 5.4 que es menor a la 5.6**

```
/**
 * @brief
 *
 */
// Si el kernel es menor al 5.6 usan file_operations
You, 3 days ago | 1 author (You)
static struct file_operations operaciones =
    {
        .open = al_abrir,
        .read = seq_read};
/**
```