Universidad Autónoma de Nicaragua

UNAN-León

Facultad de ciencias y tecnología

Departamento de computación

Ingeniería telemática

10 de may. de 21

Docente:

MSc. Wilmer Matamoros

Componente:

Gestión de red

Practica 3 Agente SNMP

Elaborado por:

Br. Johnny Jesús Andino González

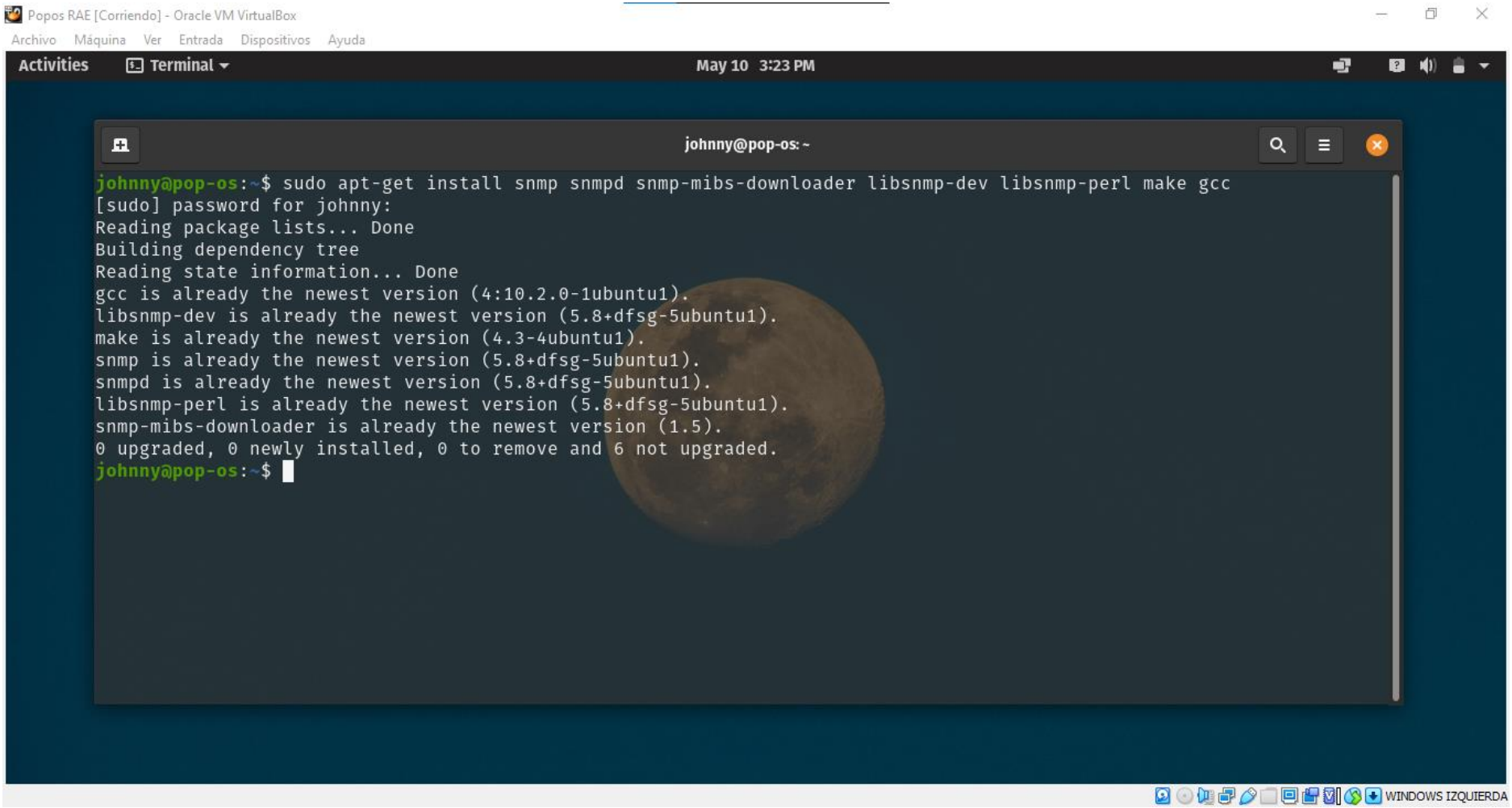
15-15530-7

Johnnyandino.15@est.unanleon.edu.ni

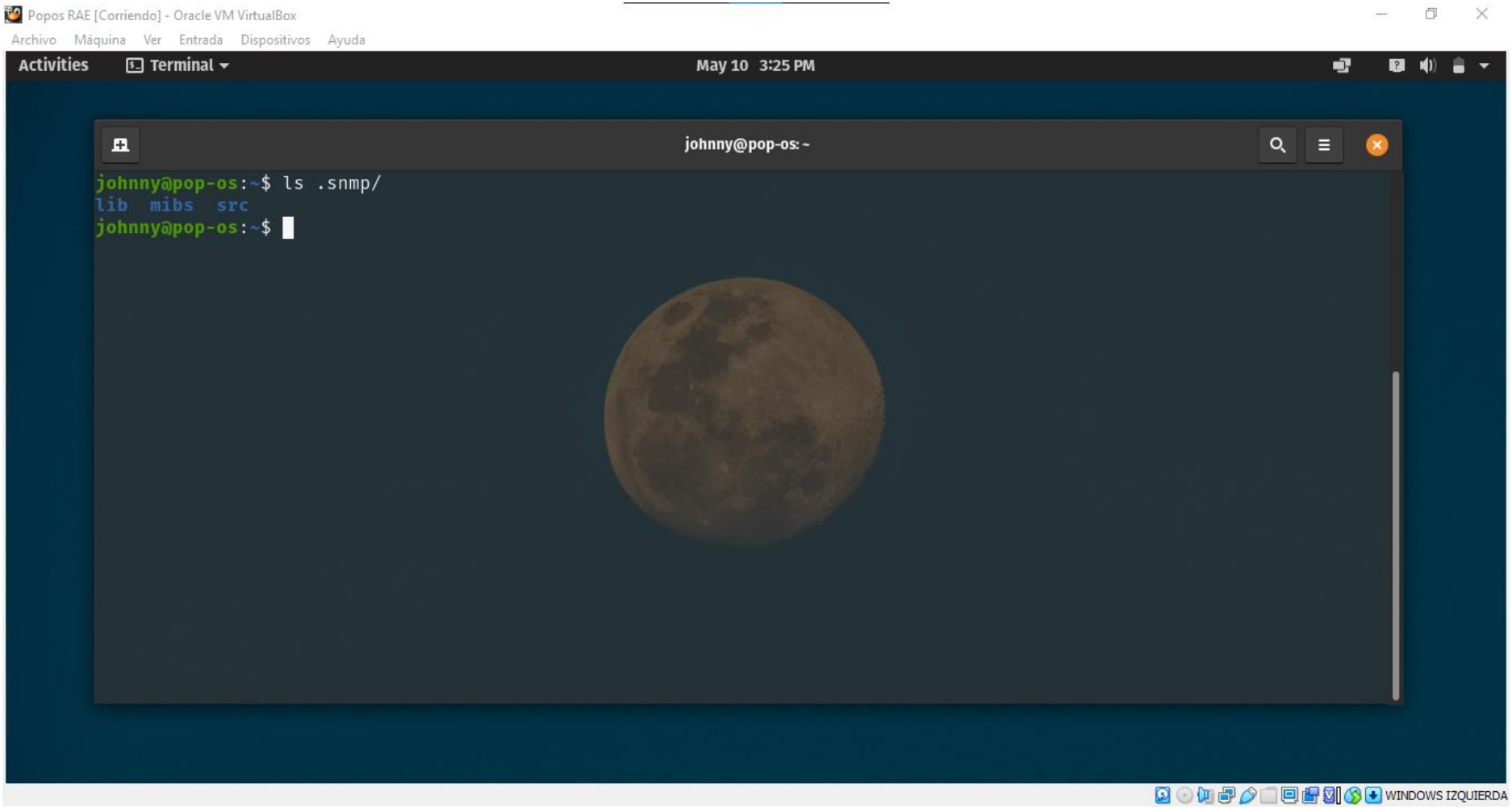
Contenido

Figuras

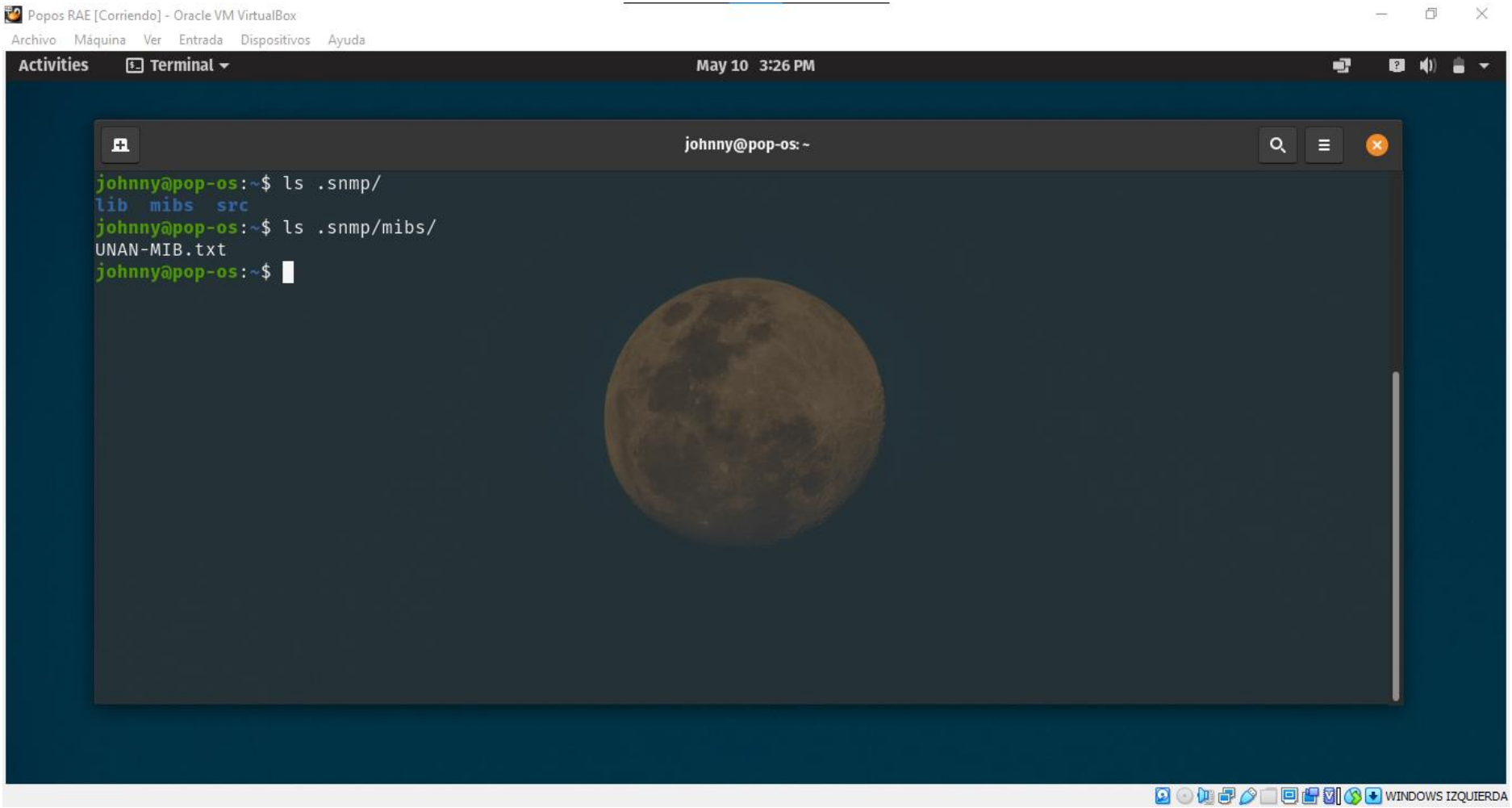
Agente SNMP



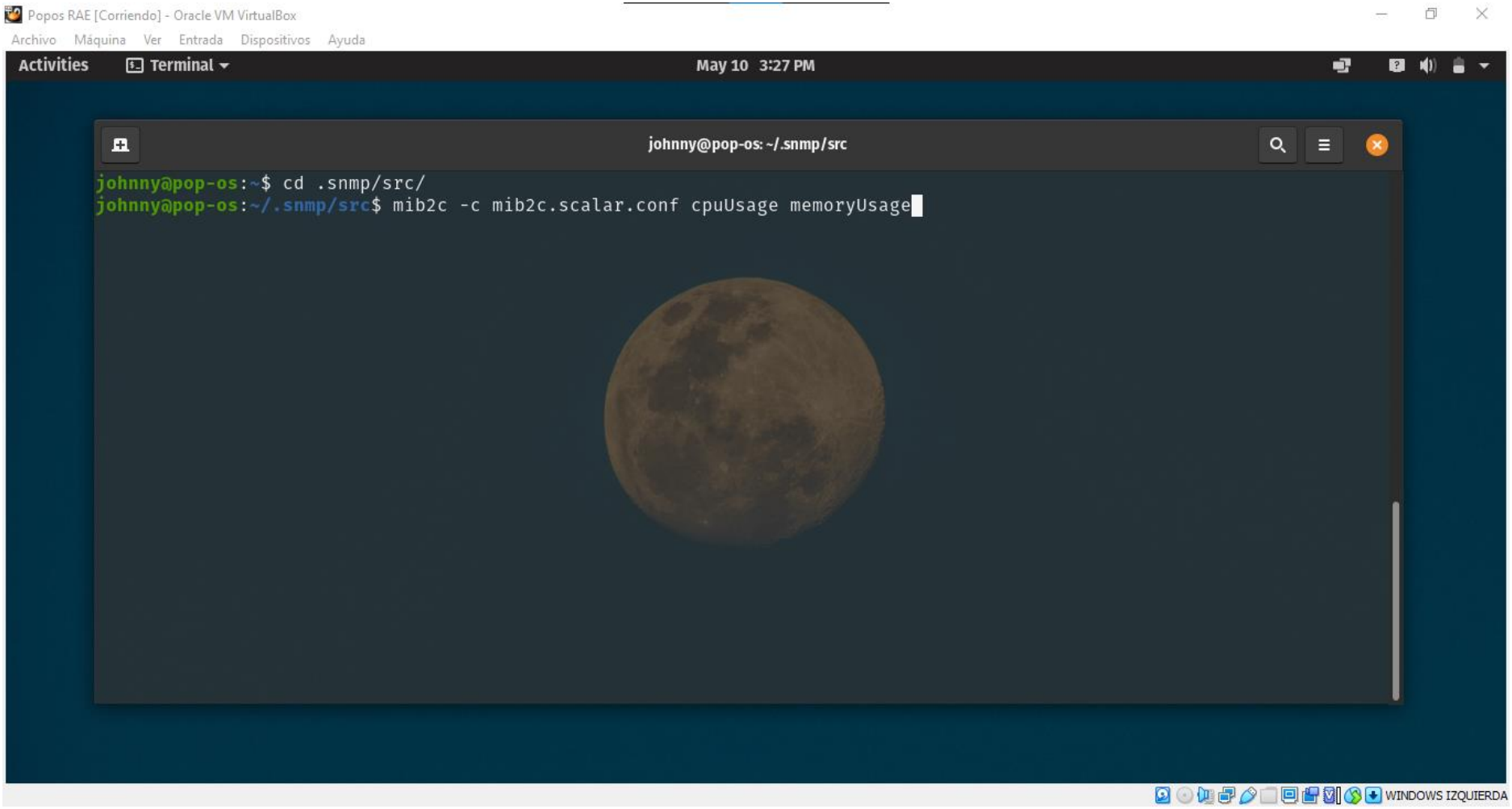
1 Instalación de paquetes necesarios para la práctica.



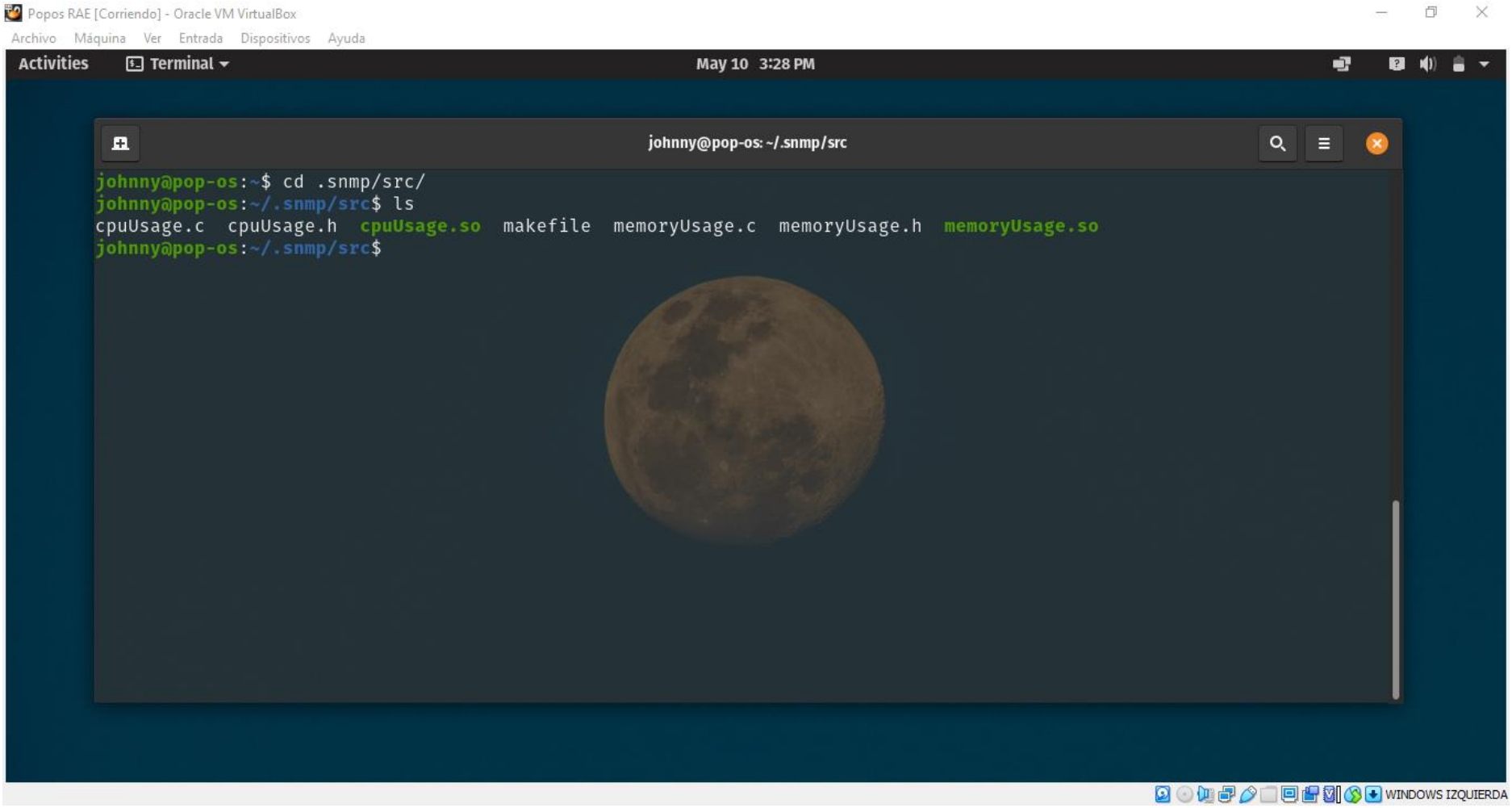
2 Creación de directorios necesarios para la practica



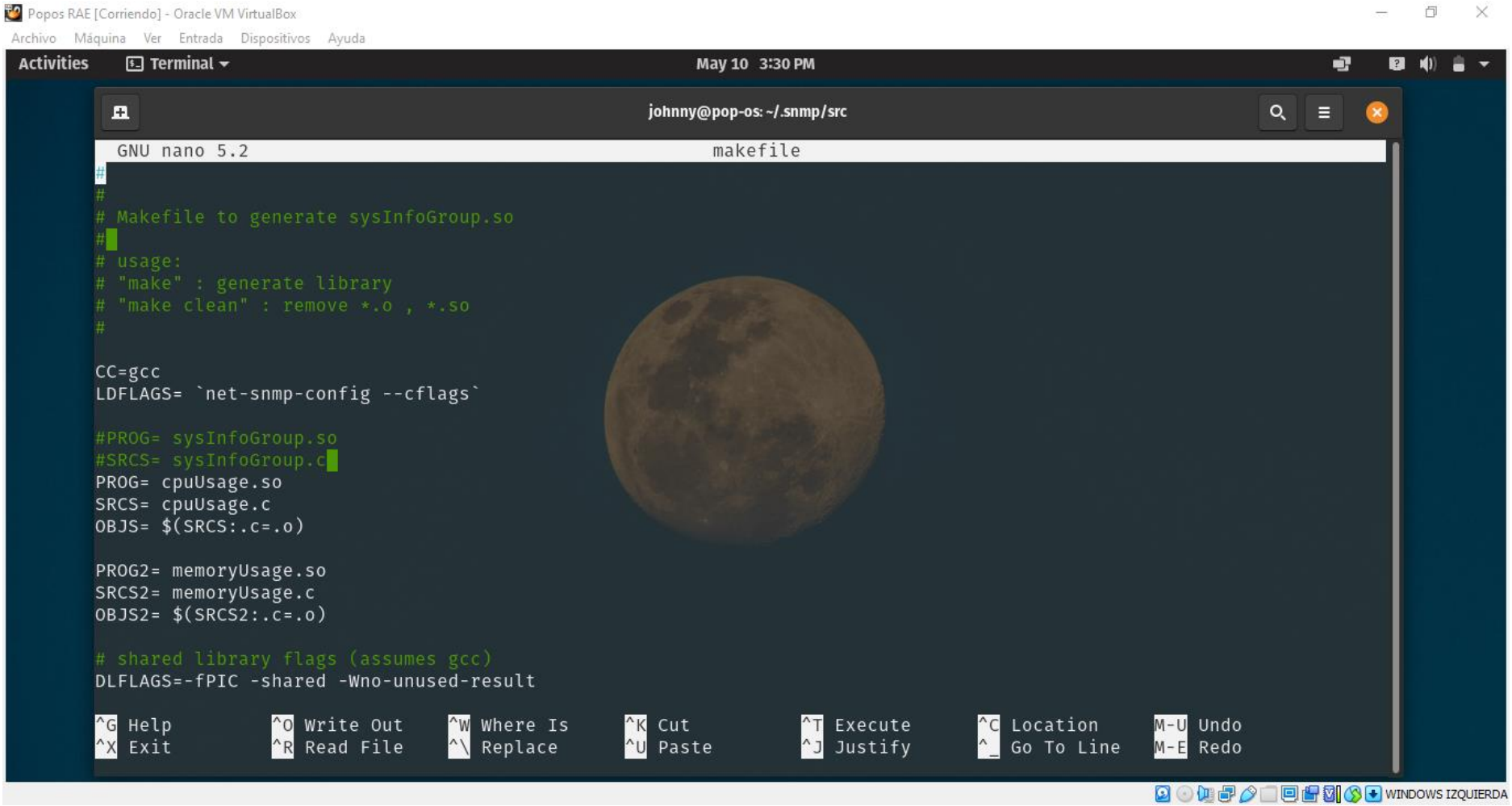
3 Creacion y configuracion del fechero donde estara la MIB (MIB completa en anexo)



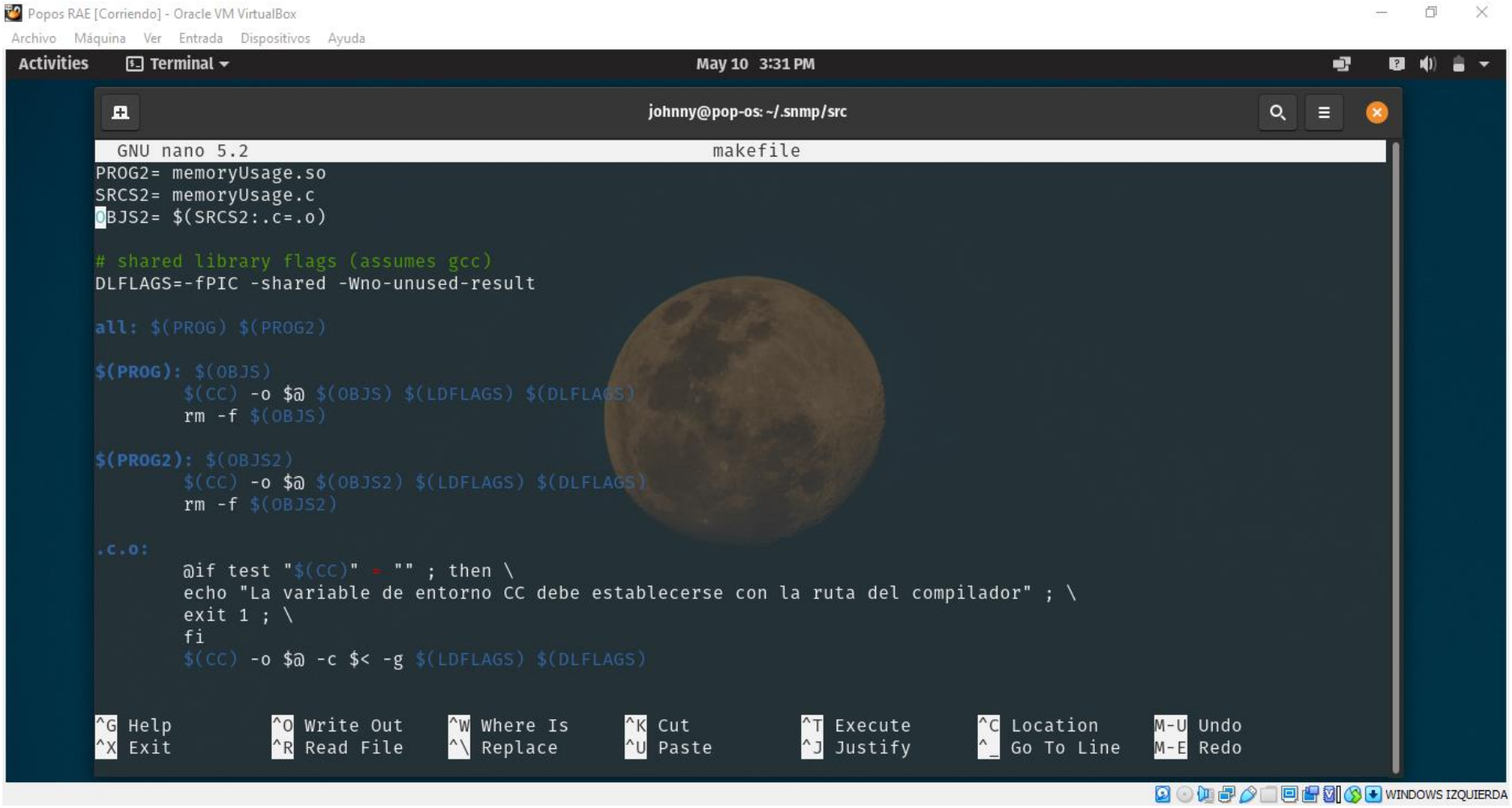
4 Comando para generar codigo de cpuUsage y memoryUsage



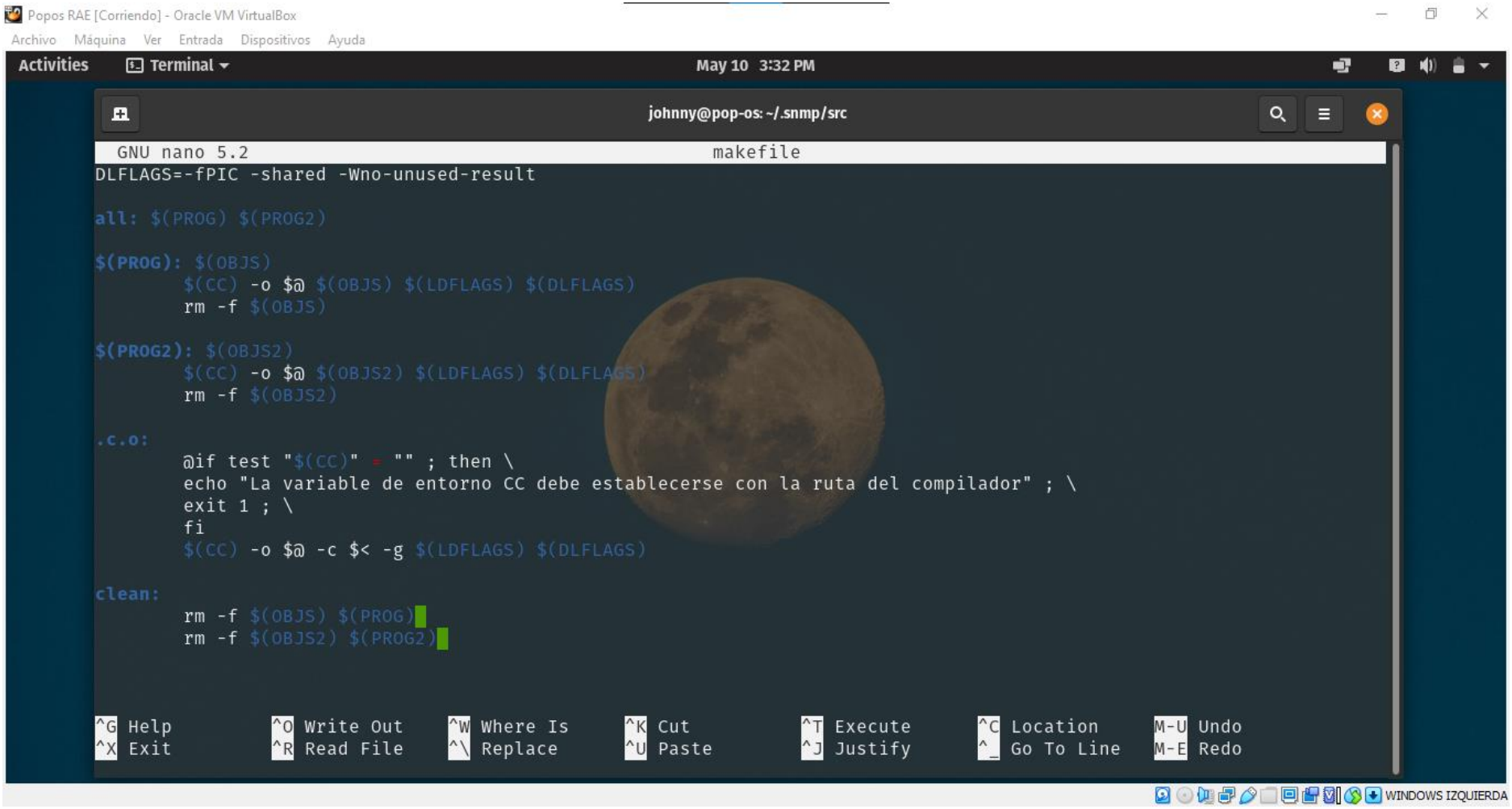
5 Resultado de la ejecución del comando anterior(Excluyendo los .so)



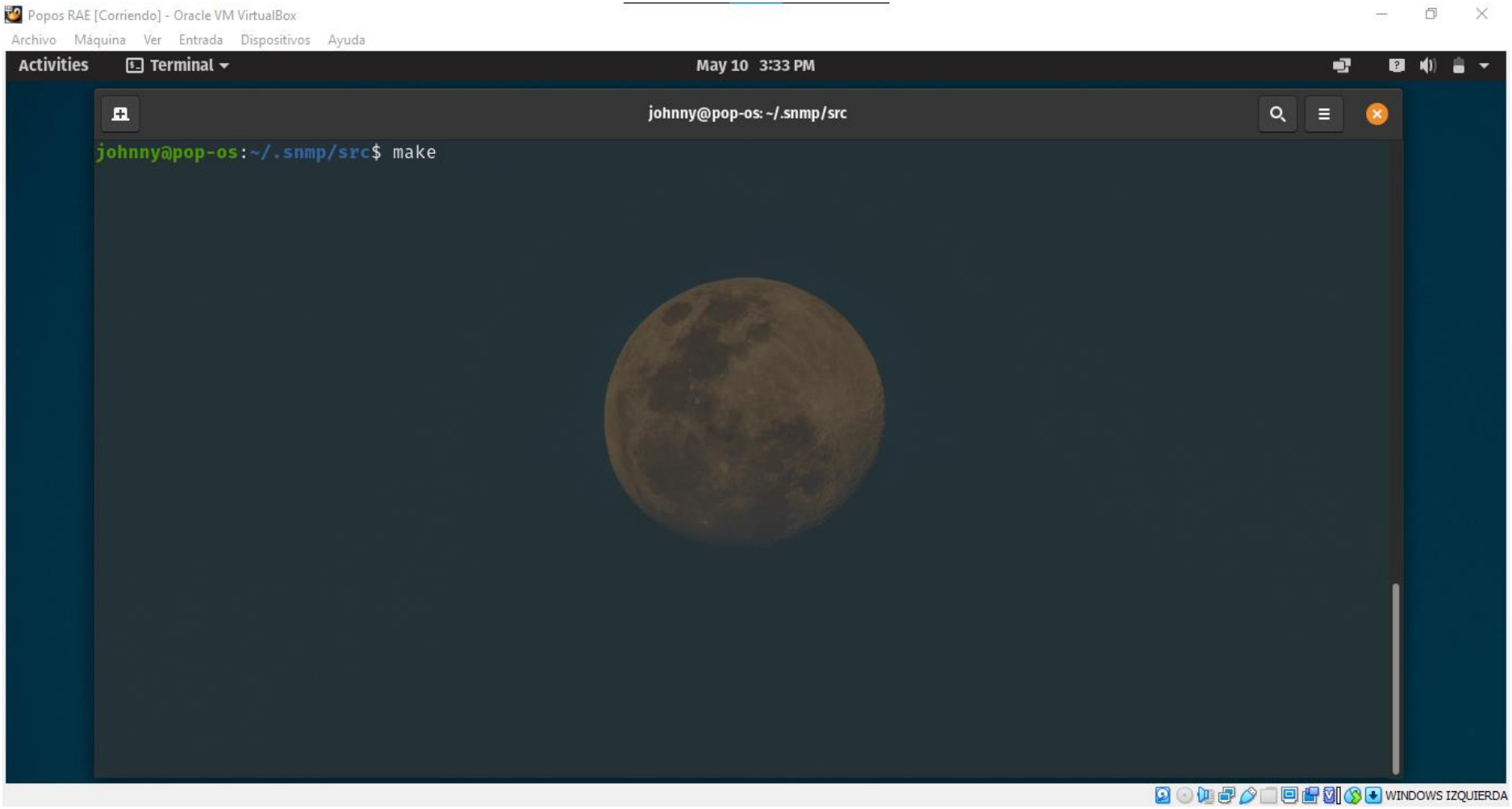
6 Configuración del fichero makenfile 1 para generar los .so



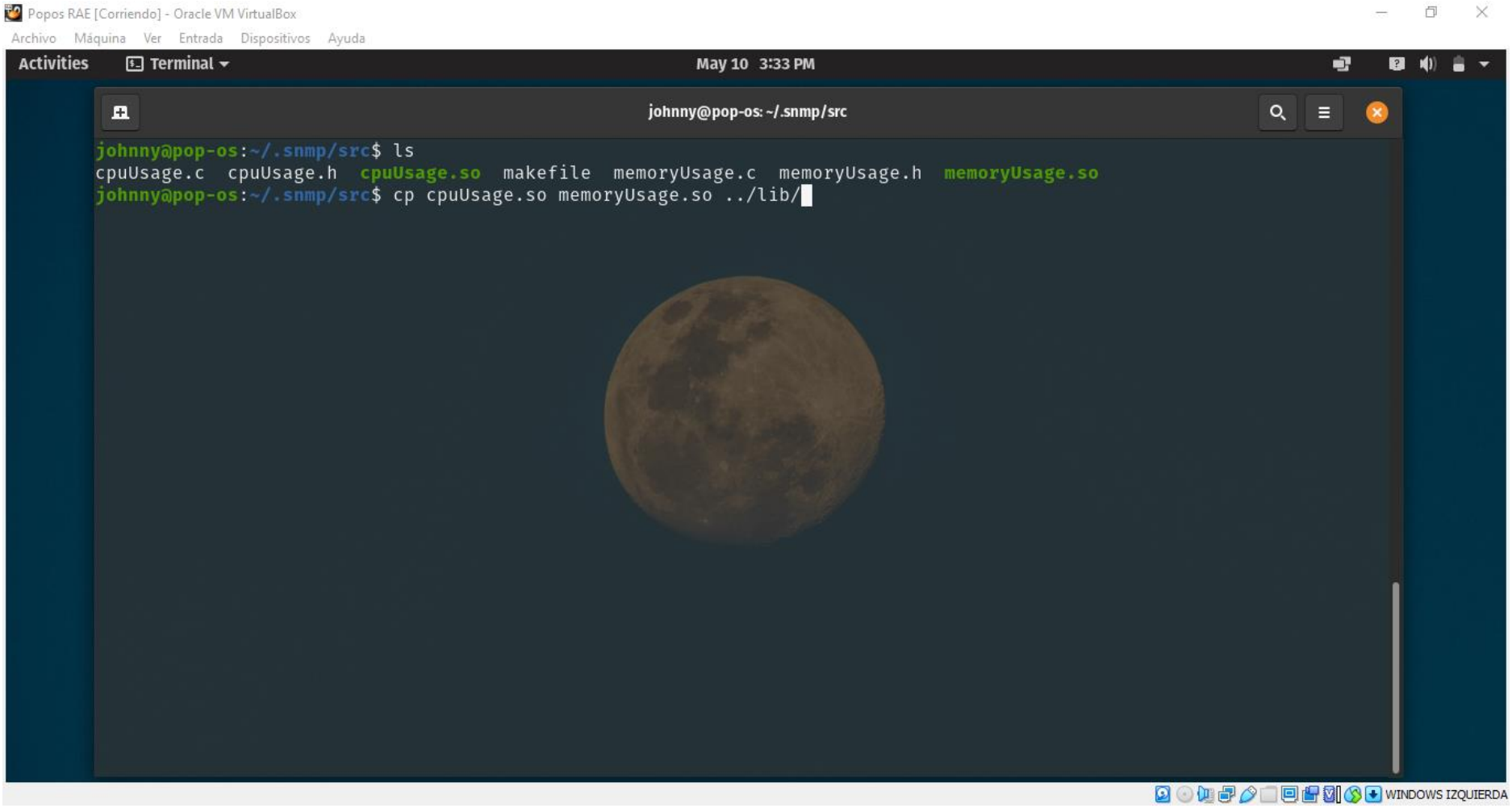
7 Configuración de makenfile 2



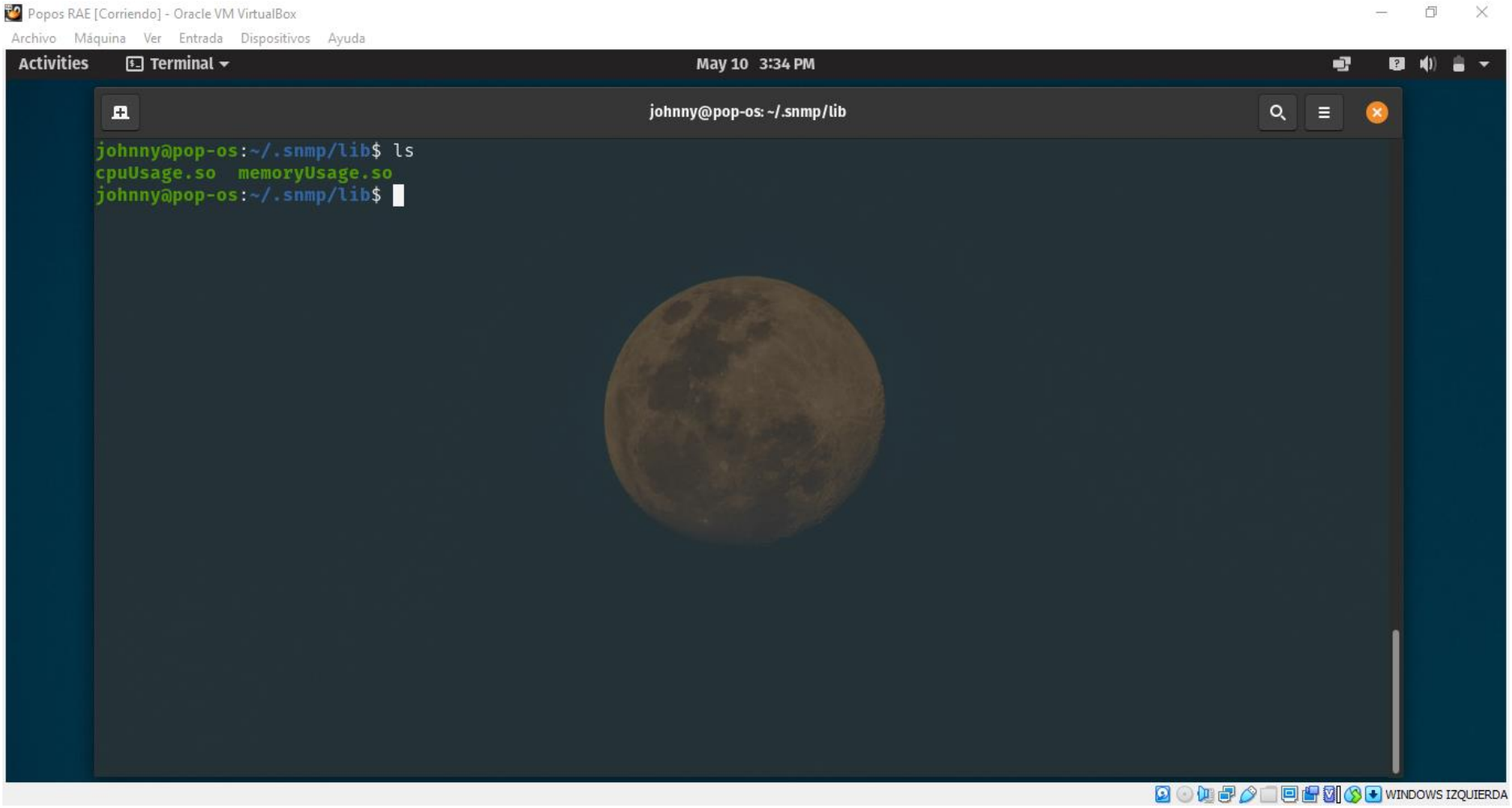
8 Fin de configuración de makenfile



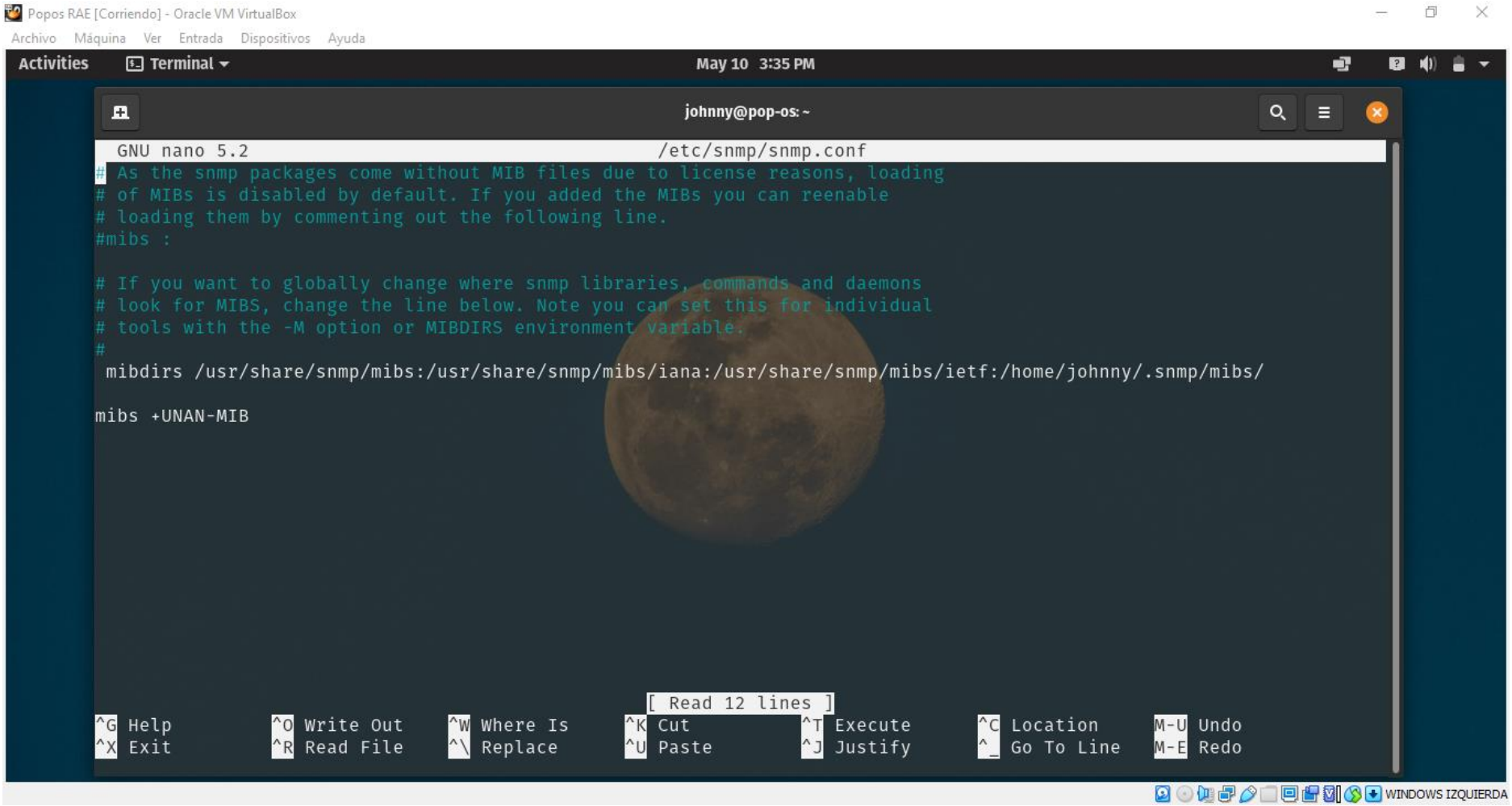
9 Comando para ejecutar los ficheros .c alojados en el directorio src para generar los .so



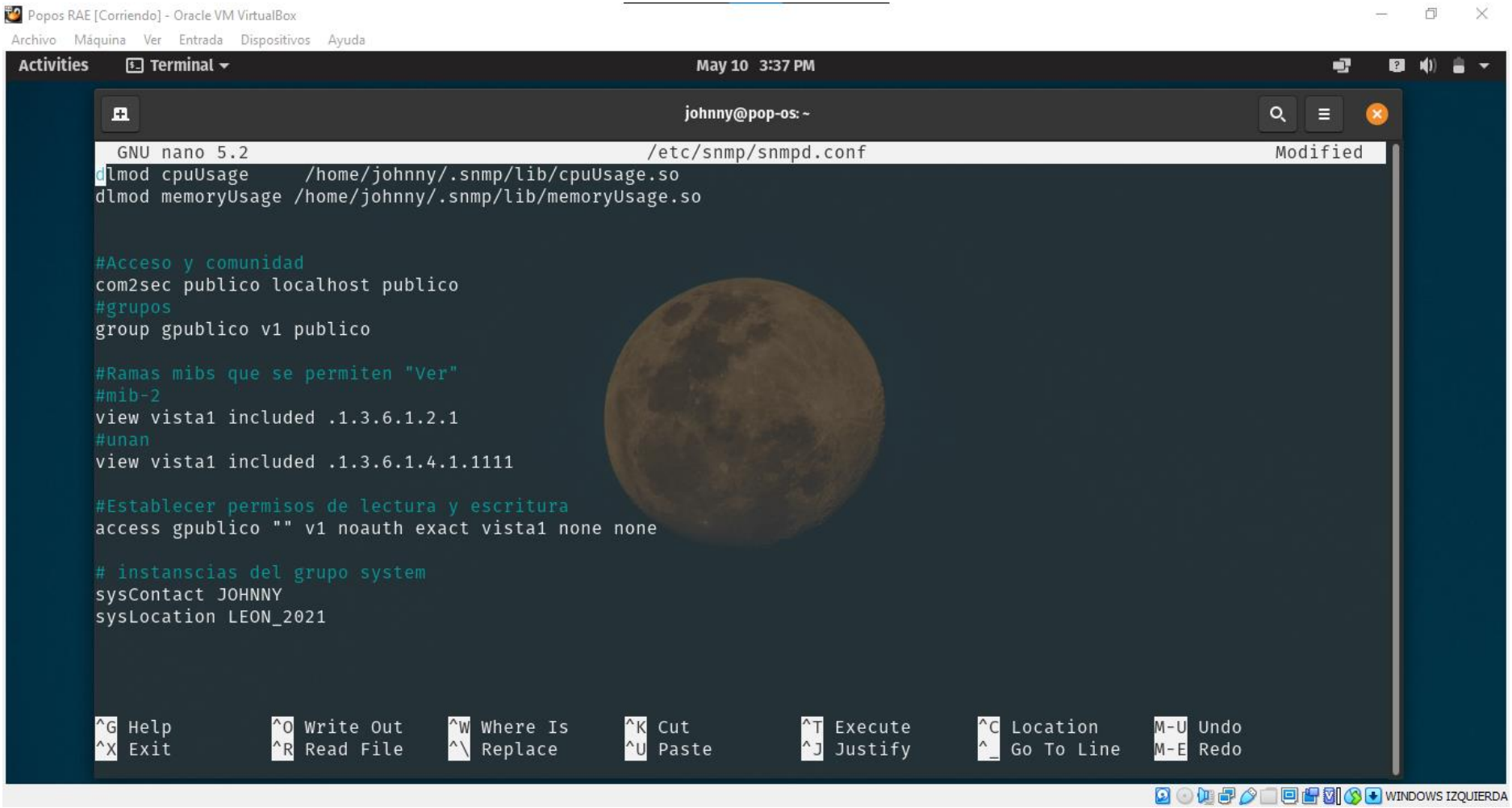
10 .so generados con el comando anterior, copiado de estos ficheros al directorio lib



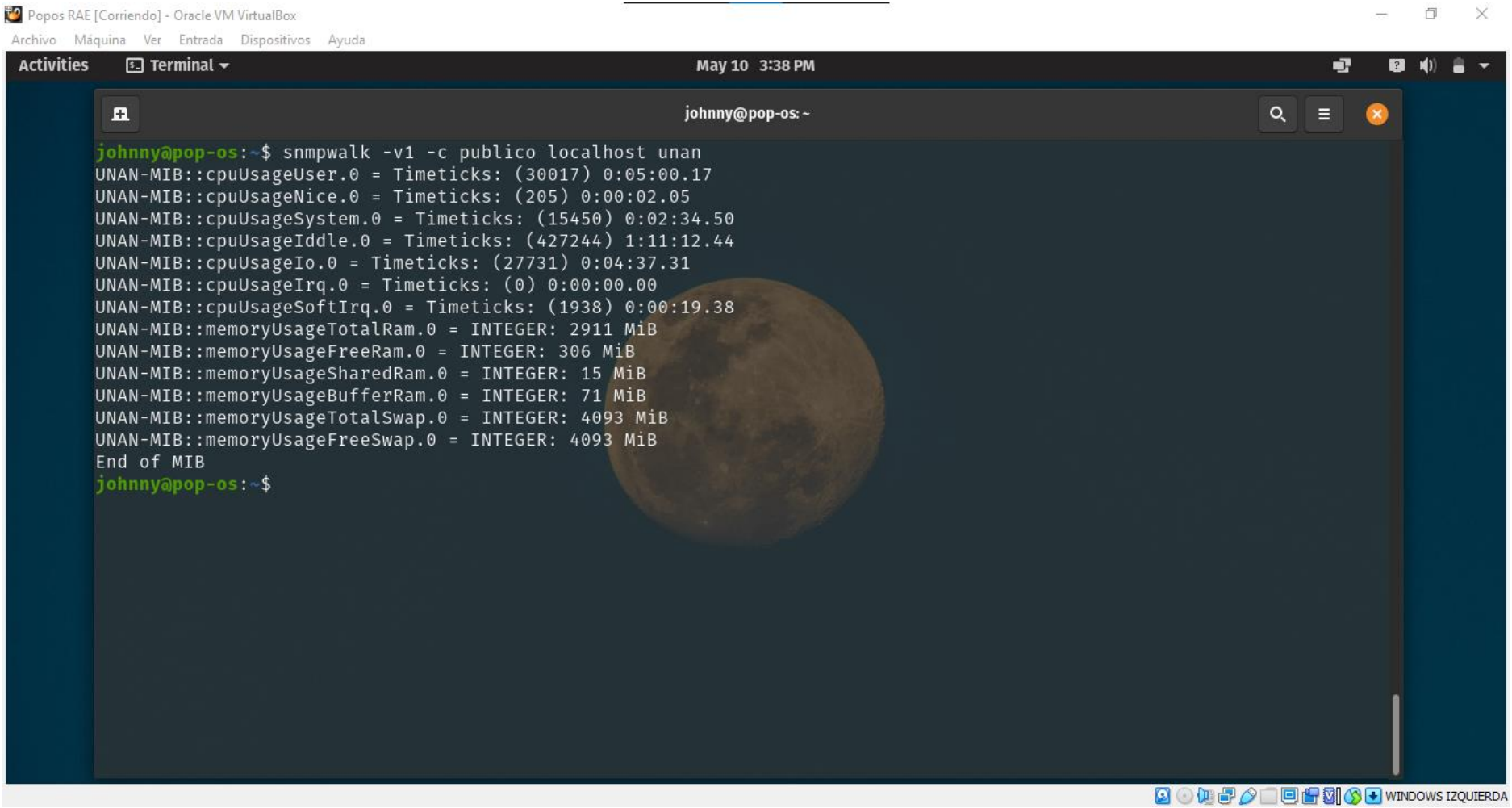
11 verificamos que se hayan copiado correctamente



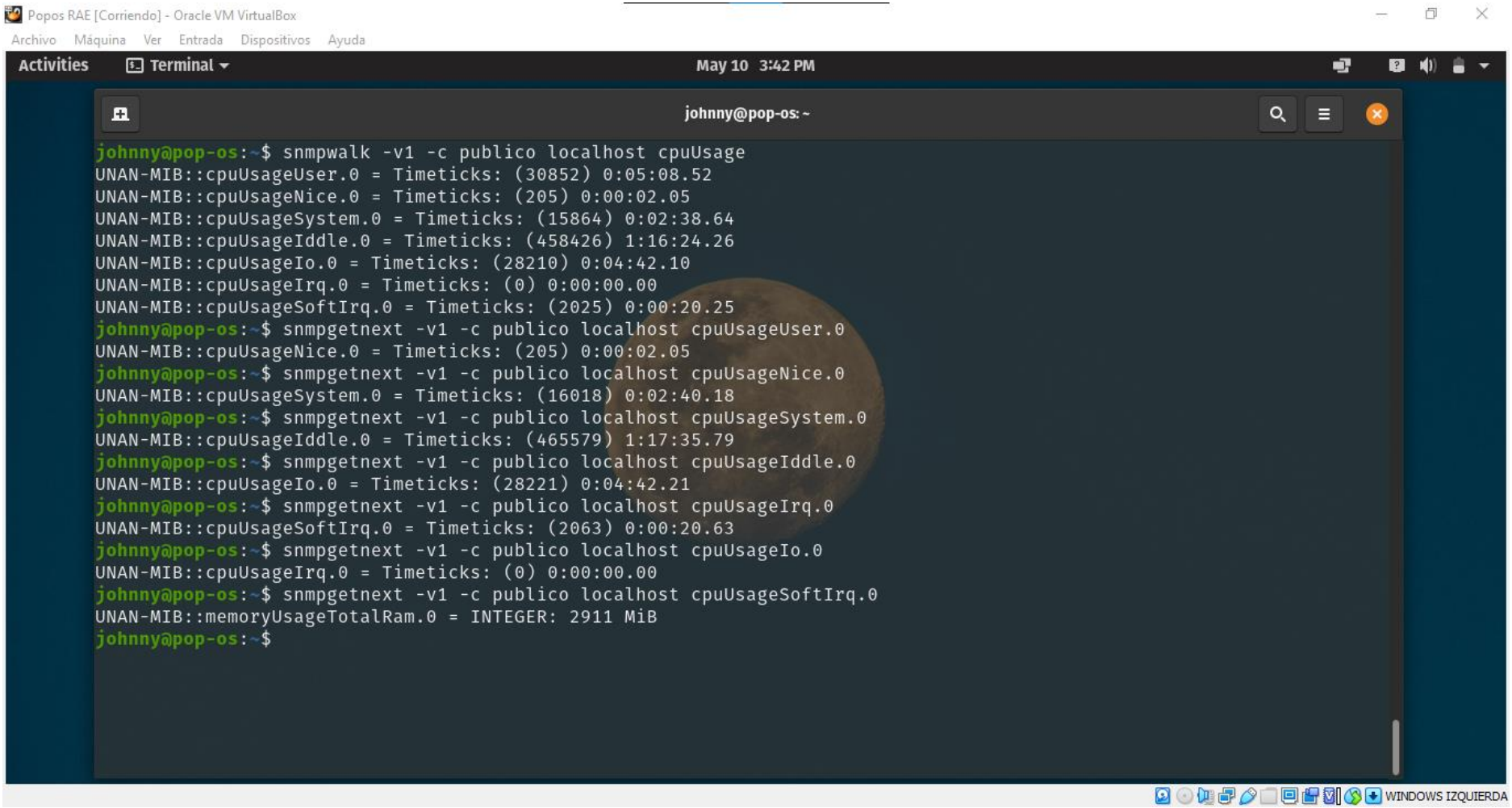
12 Configuración de snmp.conf



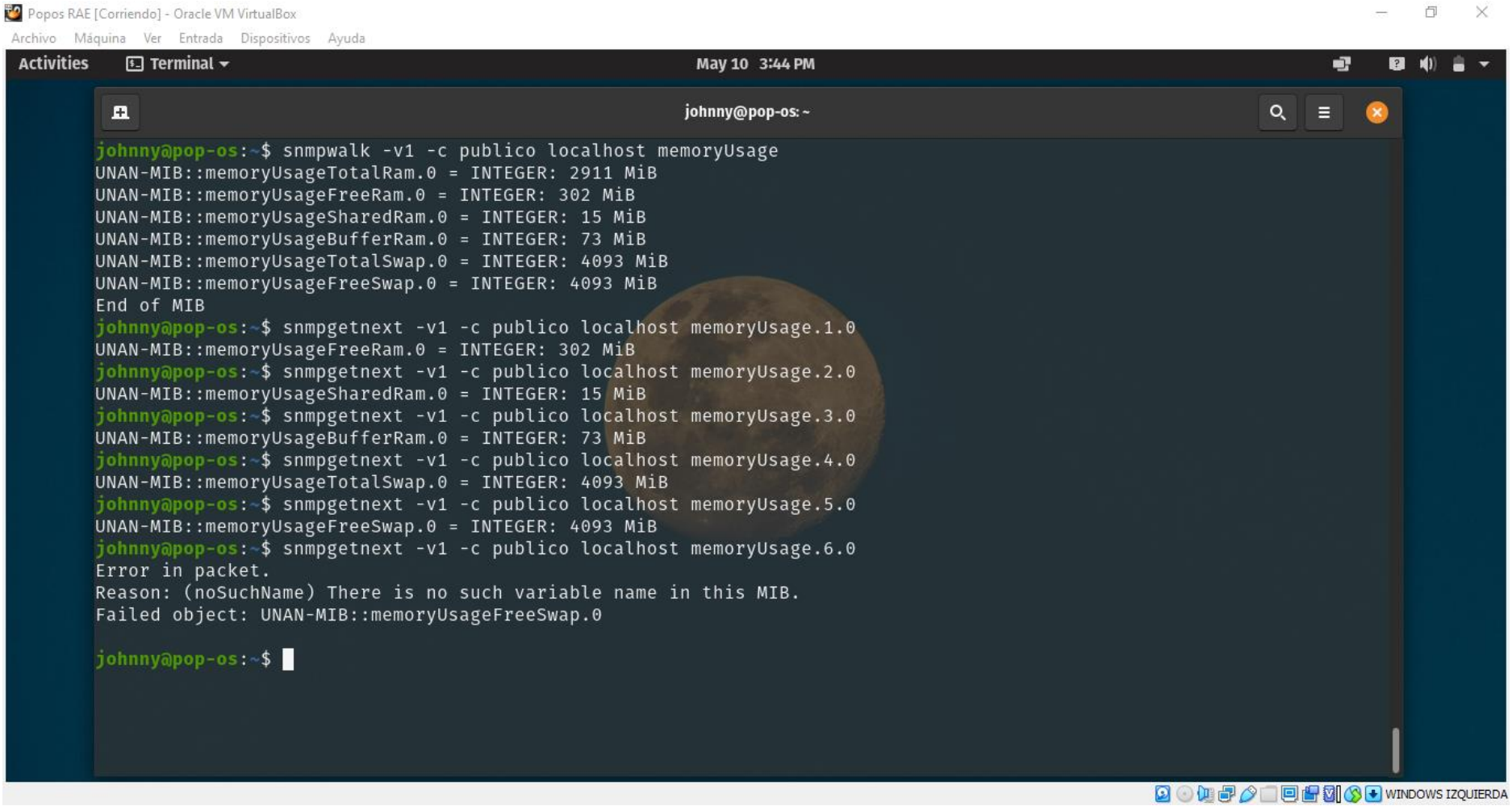
13 Configuración de las comunidades en snmpd.conf



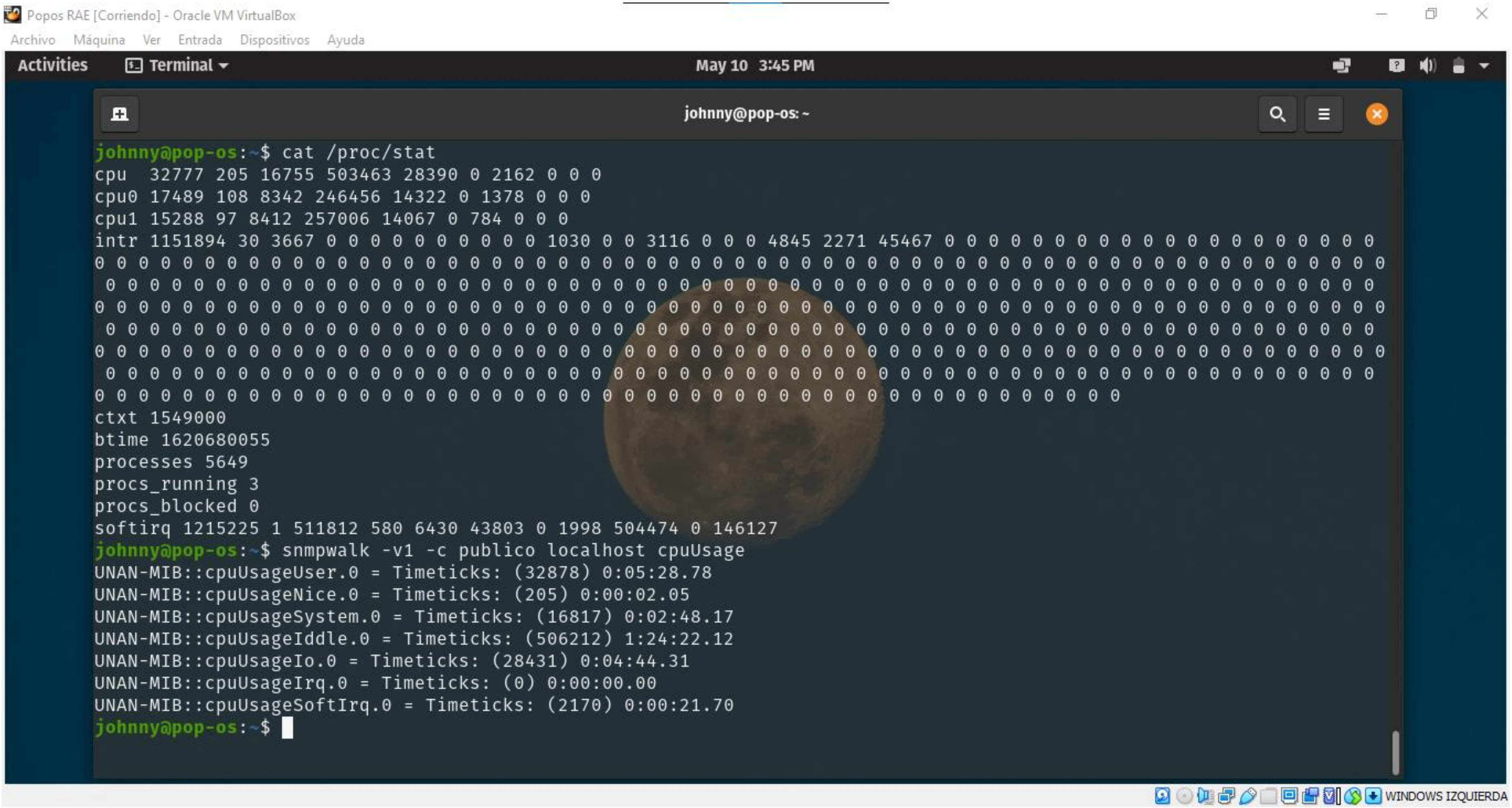
14 Prueba de funcionamiento de la mib unan con snmpwalk



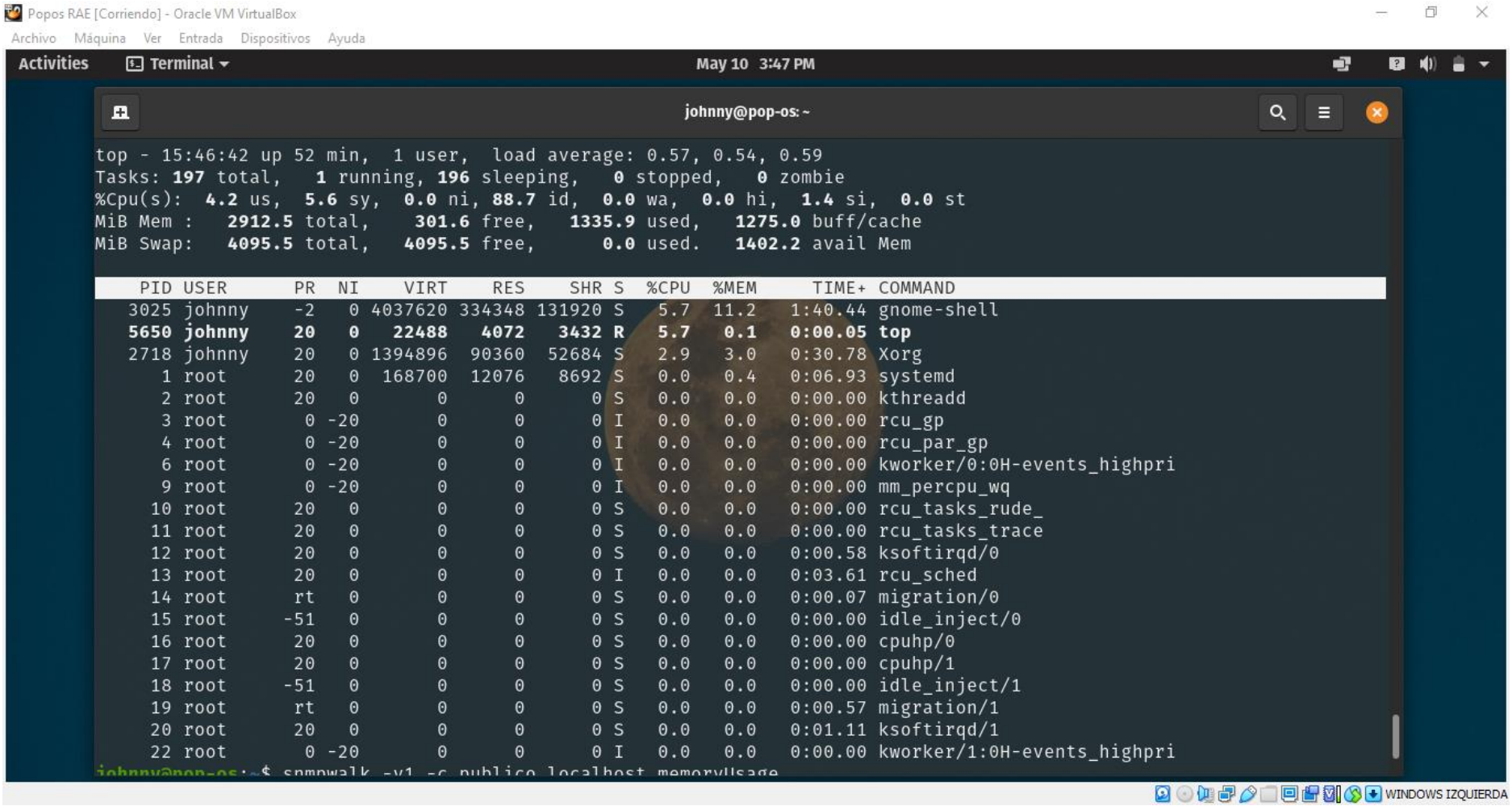
15 Prueba de funcionamiento usando snmpwalk y snmpgetnext en mib cpuUsage



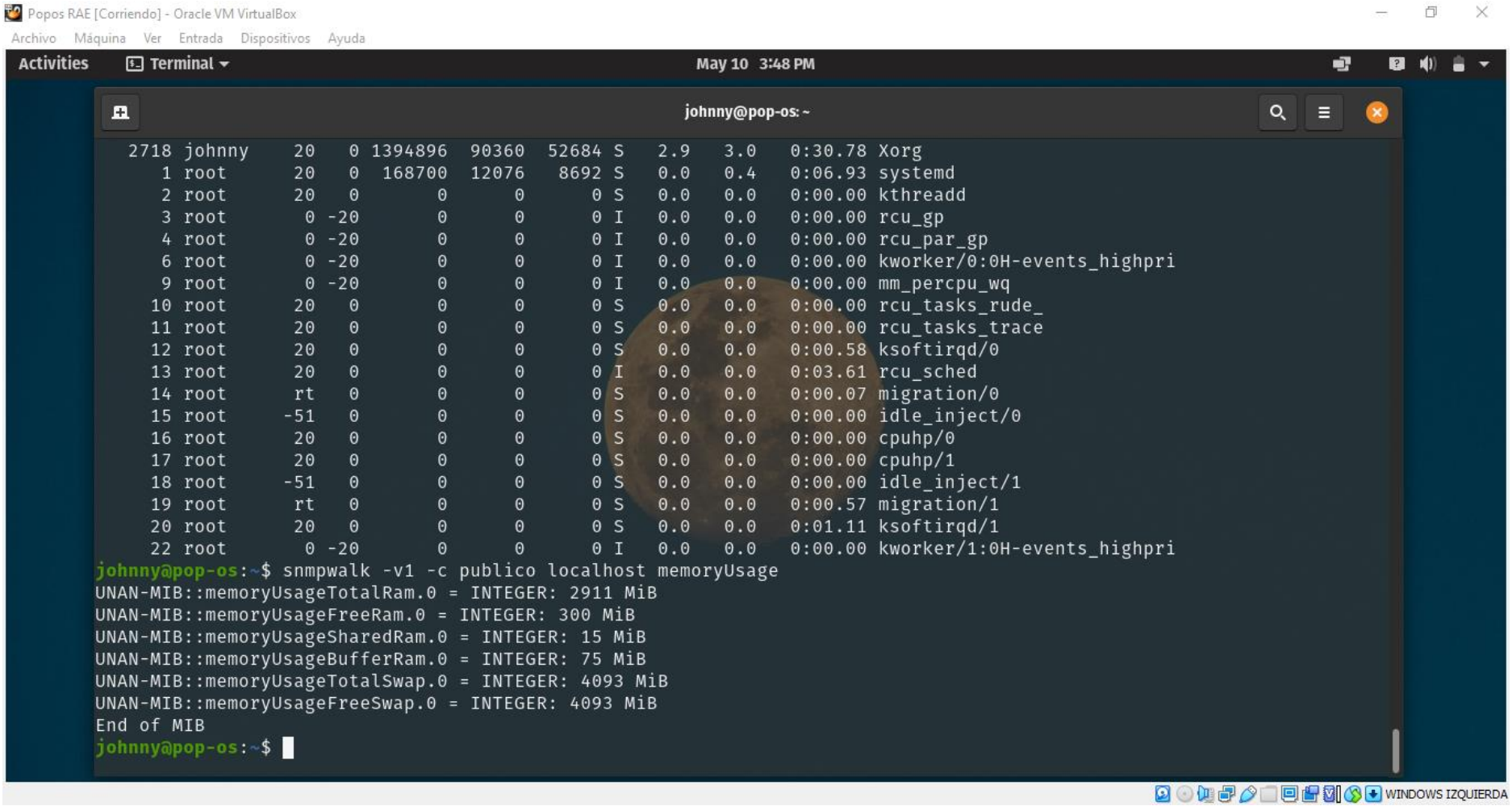
16 Prueba de fucnionamiento de mib memoryUsage con snmpwalk y snmpgetnext



17 Comparación de resultados obtenidos al ejecutar el comando proc/stat y los resultados con la mib cpuUsage



18 Comparación del comando top y mib memoryUsage 1



19 Comparación del comando top y mib memoryUsage 2

ANEXO

Enlace de video demostrativo

https://youtu.be/xixlSjYmRvA

Recursos disponibles en GitHub

https://github.com/JonesoO/gestion-de-red

MIB UNAN

```
UNAN-MIB DEFINITIONS ::= BEGIN

IMPORTS
        DisplayString
         FROM RFC1213-MIB
    enterprises, TimeTicks, Gauge, INTEGER
         FROM RFC1155-SMI
    OBJECT-TYPE
         FROM RFC-1212;

unan          OBJECT IDENTIFIER ::= { enterprises 1111 }
resources     OBJECT IDENTIFIER ::= { unan 1 }
cpuUsage      OBJECT IDENTIFIER ::= { resources 1 }
memoryUsage   OBJECT IDENTIFIER ::= { resources 2 }



cpuUsageUser   OBJECT-TYPE
        SYNTAX     TimeTicks
        ACCESS     read-only
        STATUS     mandatory
        DESCRIPTION
                "Consumo de cpu de usuario"
```

::= { cpuUsage 1 }

cpuUsageNice OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"Consumo de cpu de usuario de baja prioridad"

::= { cpuUsage 2 }

cpuUsageSystem OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"Consumo de cpu del sistema"

::= { cpuUsage 3 }

cpuUsageIddle OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"¿?"

::= { cpuUsage 4 }

cpuUsageIo OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"Consumo de cpu en peticiones IO"

::= { cpuUsage 5 }

cpuUsageIrq OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"Interrupciones"

::= { cpuUsage 6 }

cpuUsageSoftIrq OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"¿Soft IRQ?"

::= { cpuUsage 7 }

--- Subsystems group

memoryUsageTotalRam OBJECT-TYPE

SYNTAX INTEGER

UNITS "MiB"

ACCESS read-only

STATUS current

DESCRIPTION

"Cantidad en Mebibyte de RAM total"

::= { memoryUsage 1 }

memoryUsageFreeRam OBJECT-TYPE

SYNTAX INTEGER

UNITS "MiB"

ACCESS read-only

STATUS current

DESCRIPTION

"Cantidad en Mebibyte de RAM libre"

::= { memoryUsage 2 }

memoryUsageSharedRam OBJECT-TYPE

SYNTAX INTEGER

UNITS "MiB"

ACCESS read-only

STATUS current

DESCRIPTION

"Cantidad en Mebibyte de RAM compartida"

::= { memoryUsage 3 }

memoryUsageBufferRam OBJECT-TYPE

SYNTAX INTEGER

UNITS "MiB"

ACCESS read-only

STATUS current

DESCRIPTION

"Cantidad en Mebibyte de RAM Reservada"

::= { memoryUsage 4 }

memoryUsageTotalSwap OBJECT-TYPE

SYNTAX INTEGER

UNITS "MiB"

ACCESS read-only

STATUS current

```
        DESCRIPTION
            "Cantidad en Mebibyte de Swap total"
    ::= { memoryUsage 5 }

memoryUsageFreeSwap OBJECT-TYPE
        SYNTAX     INTEGER
        UNITS      "MiB"
        ACCESS     read-only
        STATUS     current
        DESCRIPTION
            "Cantidad en Mebibyte de Swap libre"
    ::= { memoryUsage 6 }

END
```

memoryUsage.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "memoryUsage.h"
#include <sys/sysinfo.h>

/** Initializes the memoryUsage module */
void
init_memoryUsage(void)
{
    const oid memoryUsageTotalRam_oid[]  = { 1,3,6,1,4,1,1111,1,2,1 };
    const oid memoryUsageFreeRam_oid[]   = { 1,3,6,1,4,1,1111,1,2,2 };
    const oid memoryUsageSharedRam_oid[] = { 1,3,6,1,4,1,1111,1,2,3 };
    const oid memoryUsageBufferRam_oid[] = { 1,3,6,1,4,1,1111,1,2,4 };
    const oid memoryUsageTotalSwap_oid[] = { 1,3,6,1,4,1,1111,1,2,5 };
    const oid memoryUsageFreeSwap_oid[]  = { 1,3,6,1,4,1,1111,1,2,6 };

  DEBUGMSGTL(("memoryUsage", "Initializing\n"));

    netsnmp_register_scalar(
        netsnmp_create_handler_registration("memoryUsageTotalRam", handle
_memoryUsageTotalRam,
                                memoryUsageTotalRam_oid, OID_LENGTH(memory
UsageTotalRam_oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("memoryUsageFreeRam", handle_
memoryUsageFreeRam,
                                memoryUsageFreeRam_oid, OID_LENGTH(memoryU
sageFreeRam_oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("memoryUsageSharedRam", handl
e_memoryUsageSharedRam,
                                memoryUsageSharedRam_oid, OID_LENGTH(memor
yUsageSharedRam_oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
```

```
        netsnmp_create_handler_registration("memoryUsageBufferRam", handl
e_memoryUsageBufferRam,
                                 memoryUsageBufferRam_oid, OID_LENGTH(memor
yUsageBufferRam_oid),
                                 HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("memoryUsageTotalSwap", handl
e_memoryUsageTotalSwap,
                                 memoryUsageTotalSwap_oid, OID_LENGTH(memor
yUsageTotalSwap_oid),
                                 HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("memoryUsageFreeSwap", handle
_memoryUsageFreeSwap,
                                 memoryUsageFreeSwap_oid, OID_LENGTH(memory
UsageFreeSwap_oid),
                                 HANDLER_CAN_RONLY
        ));
}

int
handle_memoryUsageTotalRam(netsnmp_mib_handler *handler,
                           netsnmp_handler_registration *reginfo,
                           netsnmp_agent_request_info   *reqinfo,
                           netsnmp_request_info         *requests)
{
    //Declarando la estructura necesaria para almacenar
    //la informacion del sistema.
    struct sysinfo info;
    long result;

    switch(reqinfo->mode) {

        case MODE_GET:
            //Obteniendo informacion del sistema y guardandola
            //en la variable especificada.
            sysinfo(&info);
            result = info.totalram / 1.049e+6;

            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     //Direccion de memoria.
                                     (u_char*) &result,
                                     //Tamaño de los datos.
                                     sizeof(info.totalram));
            break;

        default:
```

```
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_memoryUsageTot
alRam\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_memoryUsageFreeRam(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    struct sysinfo info;
    long result;

    switch(reqinfo->mode) {

        case MODE_GET:
            sysinfo(&info);
            result = info.freeram / 1.049e+6;

            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     (u_char*) &result,
                                     sizeof(info.freeram));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_memoryUsageFre
eRam\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_memoryUsageSharedRam(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    struct sysinfo info;
    long result;

    switch(reqinfo->mode) {

        case MODE_GET:
            sysinfo(&info);
```

```
            result = info.sharedram / 1.049e+6; //convirtiendo de Byte a
MibiByte

            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     (u_char*) &result,
                                     sizeof(info.sharedram));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_memoryUsageSha
redRam\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_memoryUsageBufferRam(netsnmp_mib_handler *handler,
                            netsnmp_handler_registration *reginfo,
                            netsnmp_agent_request_info   *reqinfo,
                            netsnmp_request_info         *requests)
{
    struct sysinfo info;
    long result;

    switch(reqinfo->mode) {

        case MODE_GET:
            sysinfo(&info);
            result = info.bufferram / 1.049e+6;

            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     (u_char*) &result,
                                     sizeof(info.bufferram));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_memoryUsageBuf
ferRam\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_memoryUsageTotalSwap(netsnmp_mib_handler *handler,
                            netsnmp_handler_registration *reginfo,
                            netsnmp_agent_request_info   *reqinfo,
                            netsnmp_request_info         *requests)
```

```
{
    struct sysinfo info;
    long result;

    switch(reqinfo->mode) {

        case MODE_GET:
            sysinfo(&info);
            result = info.totalswap / 1.049e+6;

            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     (u_char*) &result,
                                     sizeof(info.totalswap));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_memoryUsageTot
alSwap\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_memoryUsageFreeSwap(netsnmp_mib_handler *handler,
                           netsnmp_handler_registration *reginfo,
                           netsnmp_agent_request_info   *reqinfo,
                           netsnmp_request_info         *requests)
{
    struct sysinfo info;
    long result;

    switch(reqinfo->mode) {

        case MODE_GET:
            sysinfo(&info);
            result = info.freeswap / 1.049e+6;

            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     (u_char*) &result,
                                     sizeof(info.freeswap));
            break;

        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_memoryUsageFre
eSwap\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }
```

```
    return SNMP_ERR_NOERROR;
}
```

cpuUsage.c

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "cpuUsage.h"

/** Initializes the cpuUsage module */
void
init_cpuUsage(void)
{
    const oid cpuUsageUser_oid[]    = { 1,3,6,1,4,1,1111,1,1,1 };
    const oid cpuUsageNice_oid[]    = { 1,3,6,1,4,1,1111,1,1,2 };
    const oid cpuUsageSystem_oid[]  = { 1,3,6,1,4,1,1111,1,1,3 };
    const oid cpuUsageIddle_oid[]   = { 1,3,6,1,4,1,1111,1,1,4 };
    const oid cpuUsageIo_oid[]      = { 1,3,6,1,4,1,1111,1,1,5 };
    const oid cpuUsageIrq_oid[]     = { 1,3,6,1,4,1,1111,1,1,6 };
    const oid cpuUsageSoftIrq_oid[] = { 1,3,6,1,4,1,1111,1,1,7 };

  DEBUGMSGTL(("cpuUsage", "Initializing\n"));

    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageUser", handle_cpuUsa
geUser,
                                cpuUsageUser_oid, OID_LENGTH(cpuUsageUser_
oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageNice", handle_cpuUsa
geNice,
                                cpuUsageNice_oid, OID_LENGTH(cpuUsageNice_
oid),
                                HANDLER_CAN_RONLY
```

```
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageSystem", handle_cpuU
sageSystem,
                                cpuUsageSystem_oid, OID_LENGTH(cpuUsageSys
tem_oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageIddle", handle_cpuUs
ageIddle,
                                cpuUsageIddle_oid, OID_LENGTH(cpuUsageIddl
e_oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageIo", handle_cpuUsage
Io,
                                cpuUsageIo_oid, OID_LENGTH(cpuUsageIo_oid)
,
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageIrq", handle_cpuUsag
eIrq,
                                cpuUsageIrq_oid, OID_LENGTH(cpuUsageIrq_oi
d),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("cpuUsageSoftIrq", handle_cpu
UsageSoftIrq,
                                cpuUsageSoftIrq_oid, OID_LENGTH(cpuUsageSo
ftIrq_oid),
                                HANDLER_CAN_RONLY
        ));
}

int
handle_cpuUsageUser(netsnmp_mib_handler *handler,
                        netsnmp_handler_registration *reginfo,
                        netsnmp_agent_request_info   *reqinfo,
                        netsnmp_request_info         *requests)
{
    //Declarando el puntero para guardar la referencia
    //al fichero que se abrira.
    FILE *file;

    //Declarando las variables necesarias para almacenar
```

```
    //la informacion que se obtendra.
    char cpu[10];
    int num, i;

    switch(reqinfo->mode) {

        case MODE_GET:
            //Abriendo fichero con permisos de lectura
            //y almacenando la referencia en la variable.
            file = fopen("/proc/stat", "r");

            //Leyendo nombre de cpu.
            //NT: Este valor no se utiliza, pero es necesario leerlo.
            fscanf(file, "%s\t", cpu);

            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            for (i = 1; i <= 1; i++)
                fscanf(file, "%d\t", &num);

            //Cerrando el fichero..
            fclose(file);

            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageUser\n
", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_cpuUsageNice(netsnmp_mib_handler *handler,
                        netsnmp_handler_registration *reginfo,
                        netsnmp_agent_request_info   *reqinfo,
                        netsnmp_request_info         *requests)
{
    FILE *file;
    char cpu[10];
    int num, i;
```

```
    switch(reqinfo->mode) {

        case MODE_GET:
            file = fopen("/proc/stat", "r");

            fscanf(file, "%s\t", cpu);

            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            // i = 2 | cpuUsageNice.
            for (i = 1; i <= 2; i++)
                fscanf(file, "%d\t", &num);

            fclose(file);

            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageNice\n
", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_cpuUsageSystem(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    FILE *file;
    char cpu[10];
    int num, i;

    switch(reqinfo->mode) {

        case MODE_GET:
            file = fopen("/proc/stat", "r");

            fscanf(file, "%s\t", cpu);
```

```
            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            // i = 2 | cpuUsageNice.
            // i = 3 | cpuUsageSystem.
            for (i = 1; i <= 3; i++)
                fscanf(file, "%d\t", &num);

            fclose(file);

            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;

        default:
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageSystem
\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_cpuUsageIddle(netsnmp_mib_handler *handler,
                         netsnmp_handler_registration *reginfo,
                         netsnmp_agent_request_info   *reqinfo,
                         netsnmp_request_info         *requests)
{
    FILE *file;
    char cpu[10];
    int num, i;

    switch(reqinfo->mode) {

        case MODE_GET:
            file = fopen("/proc/stat", "r");

            fscanf(file, "%s\t", cpu);

            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            // i = 2 | cpuUsageNice.
            // i = 3 | cpuUsageSystem.
```

```
            // i = 4 | cpuUsageIddle.
            for (i = 1; i <= 4; i++)
                fscanf(file, "%d\t", &num);

            fclose(file);

            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;
        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageIddle\
n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_cpuUsageIo(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    FILE *file;
    char cpu[10];
    int num, i;

    switch(reqinfo->mode) {

        case MODE_GET:
            file = fopen("/proc/stat", "r");

            fscanf(file, "%s\t", cpu);

            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            // i = 2 | cpuUsageNice.
            // i = 3 | cpuUsageSystem.
            // i = 4 | cpuUsageIddle.
            // i = 5 | cpuUsageIo.
            for (i = 1; i <= 5; i++)
                fscanf(file, "%d\t", &num);
```

```
            fclose(file);

            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;

        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageIo\n",
 reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_cpuUsageIrq(netsnmp_mib_handler *handler,
                         netsnmp_handler_registration *reginfo,
                         netsnmp_agent_request_info   *reqinfo,
                         netsnmp_request_info         *requests)
{
    FILE *file;
    char cpu[10];
    int num, i;

    switch(reqinfo->mode) {

        case MODE_GET:
            file = fopen("/proc/stat", "r");

            fscanf(file, "%s\t", cpu);

            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            // i = 2 | cpuUsageNice.
            // i = 3 | cpuUsageSystem.
            // i = 4 | cpuUsageIddle.
            // i = 5 | cpuUsageIo.
            // i = 6 | cpuUsageIrq.
            for (i = 1; i <= 6; i++)
                fscanf(file, "%d\t", &num);

            fclose(file);
```

```
            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageIrq\n"
, reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_cpuUsageSoftIrq(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    FILE *file;
    char cpu[10];
    int num, i;

    switch(reqinfo->mode) {

        case MODE_GET:
            file = fopen("/proc/stat", "r");

            fscanf(file, "%s\t", cpu);

            //Leyendo hasta llegar al valor que queremos.
            //NT: Leera hasta que i sea igual numero especificado,
            // i = 1 | cpuUsageUser.
            // i = 2 | cpuUsageNice.
            // i = 3 | cpuUsageSystem.
            // i = 4 | cpuUsageIddle.
            // i = 5 | cpuUsageIo.
            // i = 6 | cpuUsageIrq.
            // i = 7 | cpuUsageSoftIrq.
            for (i = 1; i <= 7; i++)
                fscanf(file, "%d\t", &num);

            fclose(file);
```

```
            //Devolviendo datos de consulta.
            snmp_set_var_typed_value(requests->requestvb, ASN_TIMETICKS,
                                     //Direccion de memoria.
                                     (u_char*) &num,
                                     //Tamaño de los datos.
                                     sizeof(num));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_cpuUsageSoftIr
q\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
```

Makefile

```
#
#
# Makefile to generate sysInfoGroup.so
#
# usage:
# "make" : generate library
# "make clean" : remove *.o , *.so
#

CC=gcc
LDFLAGS= `net-snmp-config --cflags`

#PROG= sysInfoGroup.so
#SRCS= sysInfoGroup.c
PROG= cpuUsage.so
SRCS= cpuUsage.c
OBJS= $(SRCS:.c=.o)

PROG2= memoryUsage.so
SRCS2= memoryUsage.c
OBJS2= $(SRCS2:.c=.o)

# shared library flags (assumes gcc)
DLFLAGS=-fPIC -shared -Wno-unused-result
```

```
all: $(PROG) $(PROG2)

$(PROG): $(OBJS)
    $(CC) -o $@ $(OBJS) $(LDFLAGS) $(DLFLAGS)
    rm -f $(OBJS)

$(PROG2): $(OBJS2)
    $(CC) -o $@ $(OBJS2) $(LDFLAGS) $(DLFLAGS)
    rm -f $(OBJS2)

.c.o:
    @if test "$(CC)" = "" ; then \
    echo "La variable de entorno CC debe establecerse con la ruta del com
pilador" ; \
    exit 1 ; \
    fi
    $(CC) -o $@ -c $< -g $(LDFLAGS) $(DLFLAGS)

clean:
    rm -f $(OBJS) $(PROG)
    rm -f $(OBJS2) $(PROG2)
```