## 

## 

## **Práctica 1. Revisión.**

Sistemas operativos B.

Profesora: M.I. Ortiz Hernández Marcela.

grupo: 2402-03.

Semestre: 2019-2020/II.

Cantu Olivares Pedro de Jesus.

**21/Mayo//2020.**

**Revisión de la práctica de Archivo de Intercambio**

**Nombre del alumno: Cantu Olivares Pedro de Jesús Fecha: 21/Mayo/2020**

**Instrucciones:**

**Antes de empezar la revisión, comprimir en zip el código de la práctica 1 Archivo de intercambio que revisó en la sesión anterior, el nombre del archivo debe ser los apellidos de los integrantes.**

**Al finalizar la revisión el alumno deberá subir el documento contestado en formato PDF a la entrada correspondiente en la plataforma.**

**Duración de la actividad: 1 hora (hora de clase).**

**Descripción de la revisión:**

**El objetivo de esta actividad es comprobar que el alumno comprende el código de la práctica del archivo de intercambio. Se deberá de tener el código de la práctica que fue revisado y se debe realizar las siguientes modificaciones de forma individual:**

**a)** **Cada que se dé la orden de ejecutar un programa de prueba se debe de crear un segundo *archivo de revisión* con el nombre formado de la siguiente manera**

***nombreArchivo*RA.rev**

**donde *nombreArchivo* se obtiene de la línea de comandos y que puede ser cualquier archivo ejecutable del directorio *test*, al nombre del ejecutable se le agrega el postfijo *RA* y la extensión del archivo debe ser *.rev***

**por ejemplo, en el caso de que se ejecute halt, el segundo archivo que se genere deberá de llamarse *haltRA.rev***

***Anexar las líneas de código implementadas y especificar el nombre del archivo de la modificación***

**Archivo Modificado: ./nachos/userprog/addrespace.cc**

**Descripción:**

Primero agregué dos funciones para manipular la cadena que daría el nombre al nuevo archivo.

**Código implementado:**

//funcion para encontrar el indice de la ultima ocurrencia

//de un caracter dado.

int strlo(char\* array, char caracter)

{

for(int i = strlen(array)-1 ; i>-1;i--)

{

if(array[i] == caracter)

{

return i;

}

}

return -1;

}

//funcion que regresa una subcadena, dada una cadena original

//el inicio de la subcadena y su tamaño.

char\* strsub(char\* array, int indexOffset , int lenght)

{

char\* res = new char[lenght+1];

for(int i = 0; i < lenght + 1 ; i++)

{

res[i] = array[indexOffset + i] ;

}

return res;

}

**Archivo Modificado: ./nachos/userprog/addrespace.cc**

**Descripción:**

En la práctica 1 se modificó el constructor de la clase AddrSpace para pasarle el nombre por parámetro ya que se necesitaba para crear el archivo de intercambio. Aqui se dejo igual y con las funciones anteriores se lograba obtener el nombre del programa en ejecución, sin la dirección absoluta, para así poder guardarlo donde se indica.

**Código implementado:**

AddrSpace::AddrSpace(OpenFile \*executable, char\* filename)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Practica1 Revision

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int indexRealName = strlo(filename,'/') +1 ;

int realNameLength = (strlen(filename) - indexRealName +1);

char \*fileRealName = strsub(filename,indexRealName, strlen(filename) - indexRealName);

char archivoRevisionPath[11 + strlen(fileRealName) + 6] = "";

strcat(archivoRevisionPath, "../userprog/");

strcat(archivoRevisionPath, fileRealName);

strcat(archivoRevisionPath, "RA.rev");

printf("Se creara el archivo: %s\n", archivoRevisionPath);

if(!fileSystem->Create(archivoRevisionPath,executable->Length()-40))

{

printf("\nNo se pudo crear el archivo de intercambio %s\n", archivoRevisionPath );

}

else

{

OpenFile \*revisionOpenFile = fileSystem->Open(archivoRevisionPath);

if(revisionOpenFile == NULL)

{

printf("\nEl archivo de intercambio no existe\n");

}

else

{

char \*auxRev;

auxRev = new char[executable->Length()-40];

int auxInt1Rev = executable->ReadAt(auxRev,executable->Length()-40,40);

if(auxInt1Rev > 0)

{

int auxInt2Rev = revisionOpenFile->Write(auxRev,executable->Length()-40);

if(auxInt2Rev <= 0)

{

printf("\nNo se pudo escribir en el archivo de revision\n");

}

}

else

{

printf("\nNo se pudo hacer lectura del ejecutable\n");

}

}

}

}//AddrSpace constructor(END).

**b)** **El archivo de revisión se debe crear en el directorio *userprog*.**

***Anexar las líneas de código implementadas y especificar el nombre del archivo de la modificación***

**Archivo Modificado: ./nachos/userprog/addrespace.cc**

**Descripción:**

En el constructor **AddrSpace()** en el archivo

**Código implementado:**

char archivoRevisionPath[11 + strlen(fileRealName) + 6] = "";

strcat(archivoRevisionPath, "../userprog/");

strcat(archivoRevisionPath, fileRealName);

strcat(archivoRevisionPath, "RA.rev");

**c)** **El contenido del archivo de revisión debe ser solo el segmento de código del archivo ejecutable.**

***Anexar las líneas de código implementadas y especificar el nombre del archivo de la modificación***

**Archivo Modificado: ./nachos/userprog/addrespace.cc**

**Descripción:**

En el constructor **AddrSpace()** en el archivo

**Código implementado:**

**char \*auxRev;**

**auxRev = new char[executable->Length()-40];**

**int auxInt1Rev = executable->ReadAt(auxRev,executable->Length()-40,40);**

**if(auxInt1Rev > 0)**

**{**

**int auxInt2Rev = revisionOpenFile->Write(auxRev,executable->Length()-40);**

**if(auxInt2Rev <= 0)**

**{**

**printf("\nNo se pudo escribir en el archivo de revision\n");**

**}**

**}**

**else**

**{**

**printf("\nNo se pudo hacer lectura del ejecutable\n");**

**}**

**d)** **Explique los argumentos o parámetros que reciben las funciones *WriteAt* y *ReadAt*.**

**writeAt:**

**OpenFile :: Write ( char \* into, int numBytes);**

**into:** corresponde a la información(buffer) que se escribirá en el OpenFile, desde el que se llamó tal método.

**numBytes:** la cantidad de información que se leerá del parámetro into.

**ReadAt:**

**OpenFile:: ReadAt(char \* into, int numBytes, int position);**

**into:** el buffer a donde se guardará la información que se dará lectura.

**numBytes:** la cantidad de bytes que se leerán.

**Position:** la posición en el archivo desde la que se leerán el numero de bytes indicados.

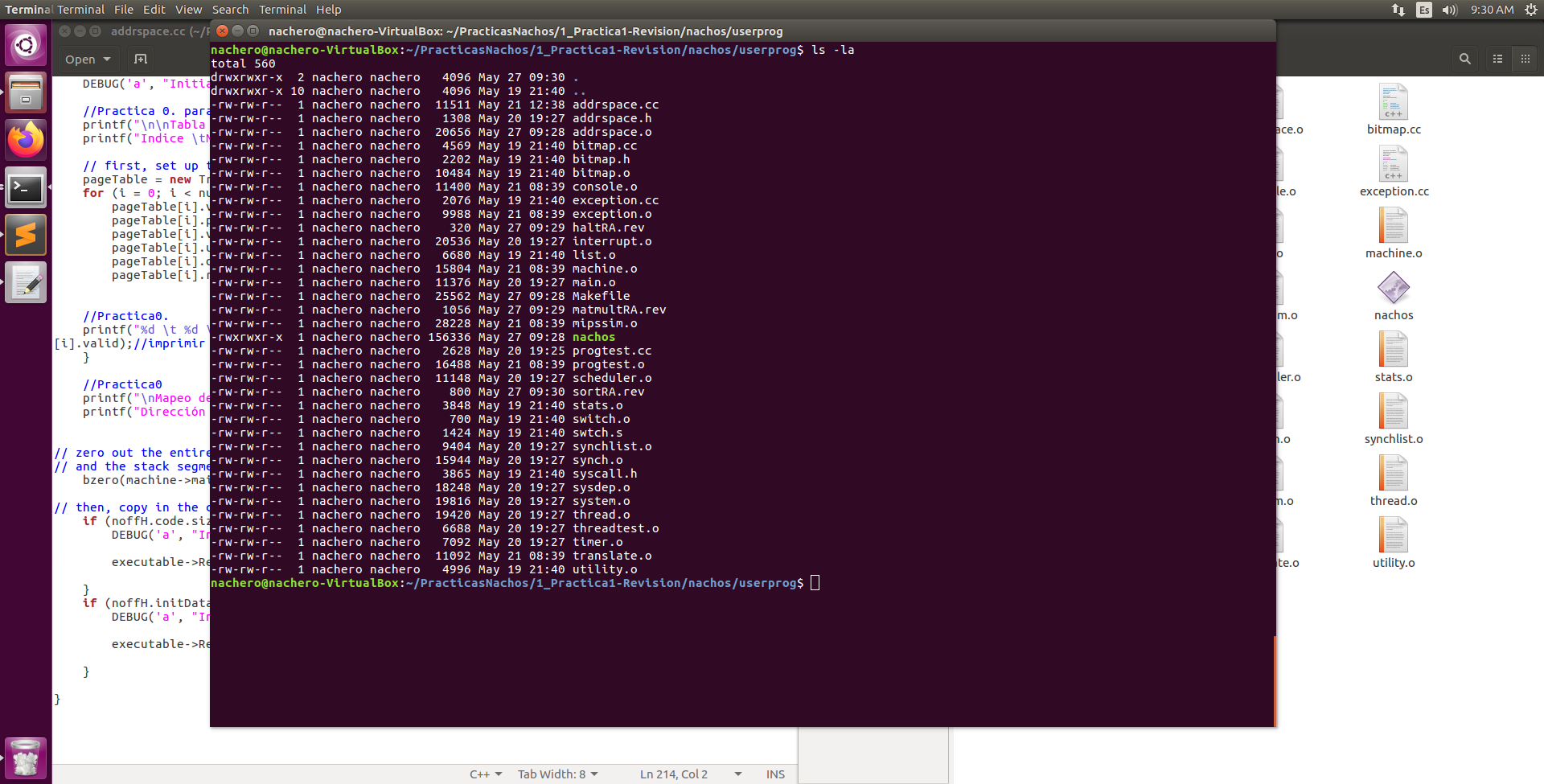
**e) El funcionamiento del administrador de memoria debe ser igual al de la práctica 0, es decir, el único programa que se ejecuta es *halt* y los demás van a marcar errores y no se ejecutarán. La diferencia es que ahora por cada programa de prueba se debe generar el archivo de intercambio y un archivo de revisión.**

## 

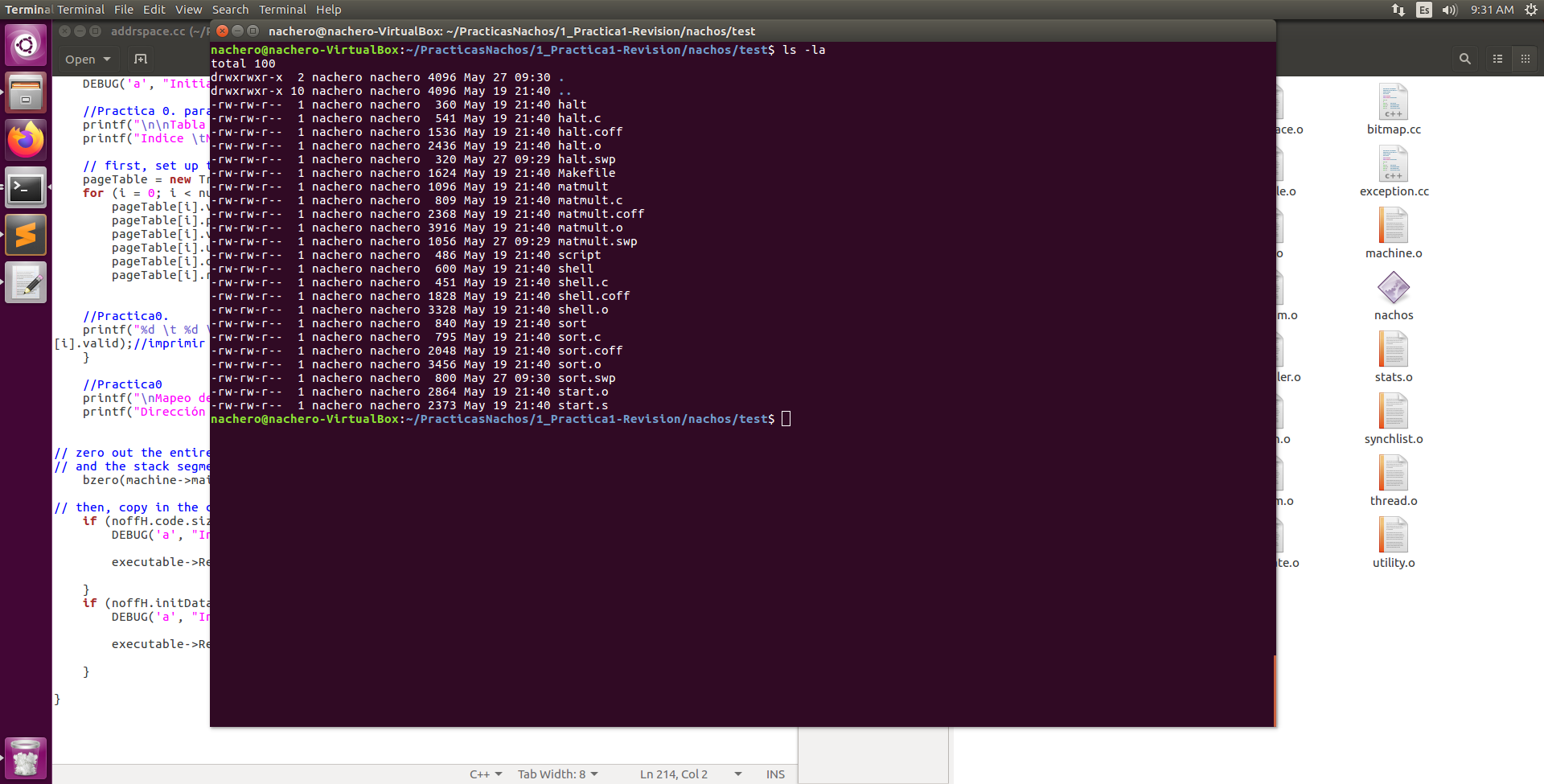
## 

**f)** **Ejecute el administrador de memoria con los programas: halt, matmult y sort**

**Liste el contenido del directorio *userprog y anexe una captura de pantalla***

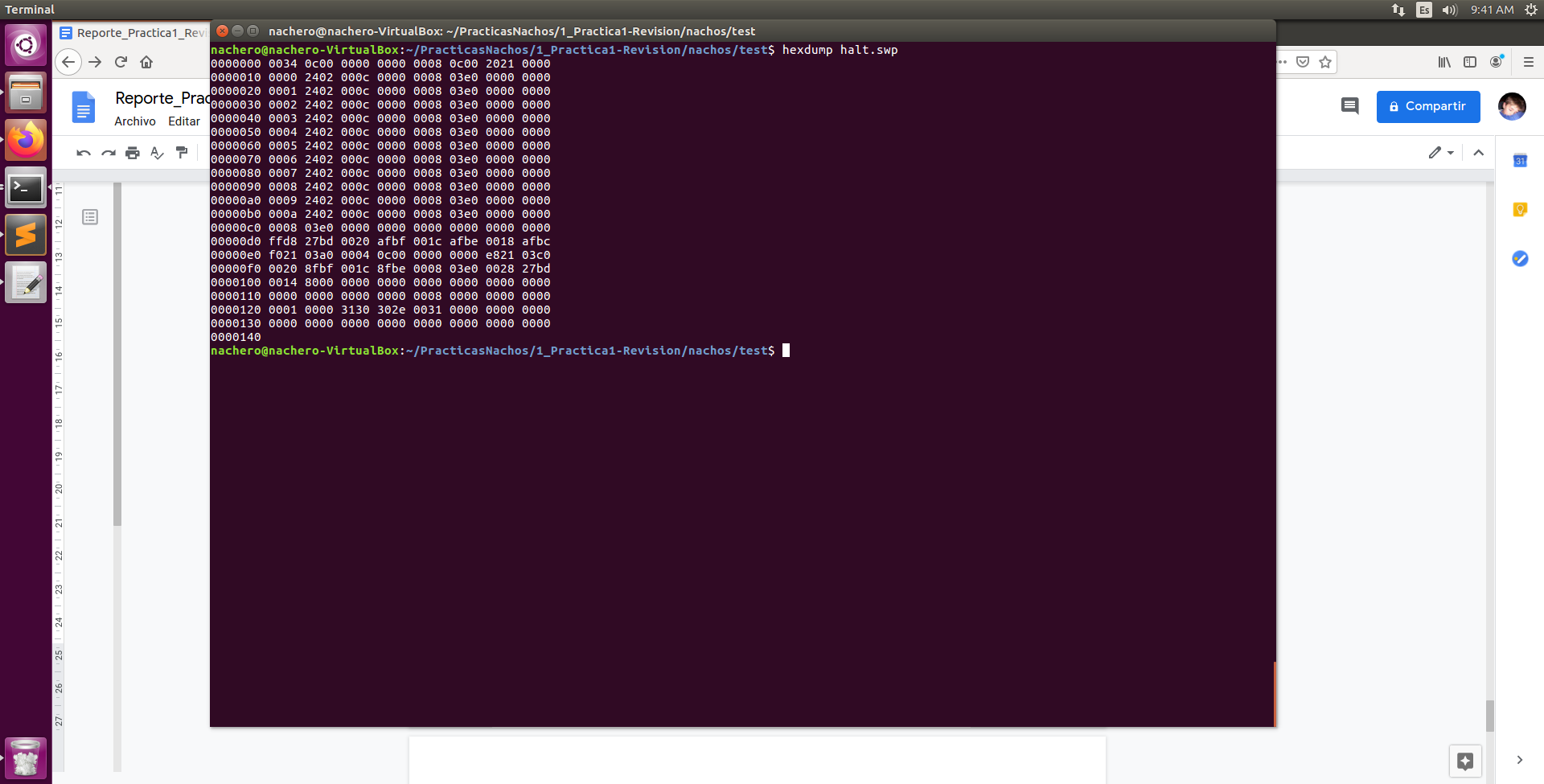
****

**Liste el contenido del directorio *test y anexe una captura de pantalla***

****

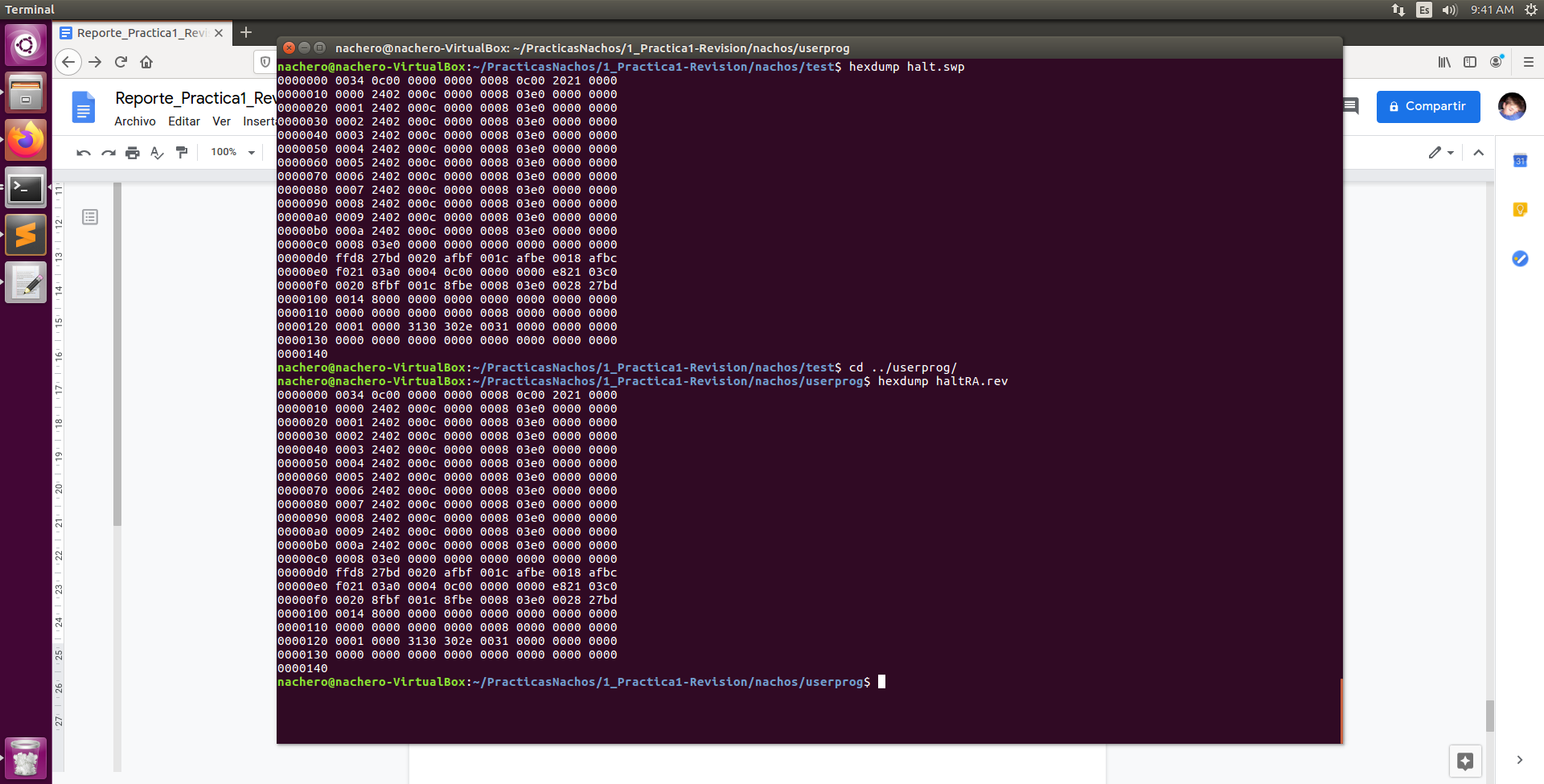
**g)** **Despliegue el contenido del archivo *halt.swp* usando el comando hexdump**

***Anexe una captura de pantalla***

****

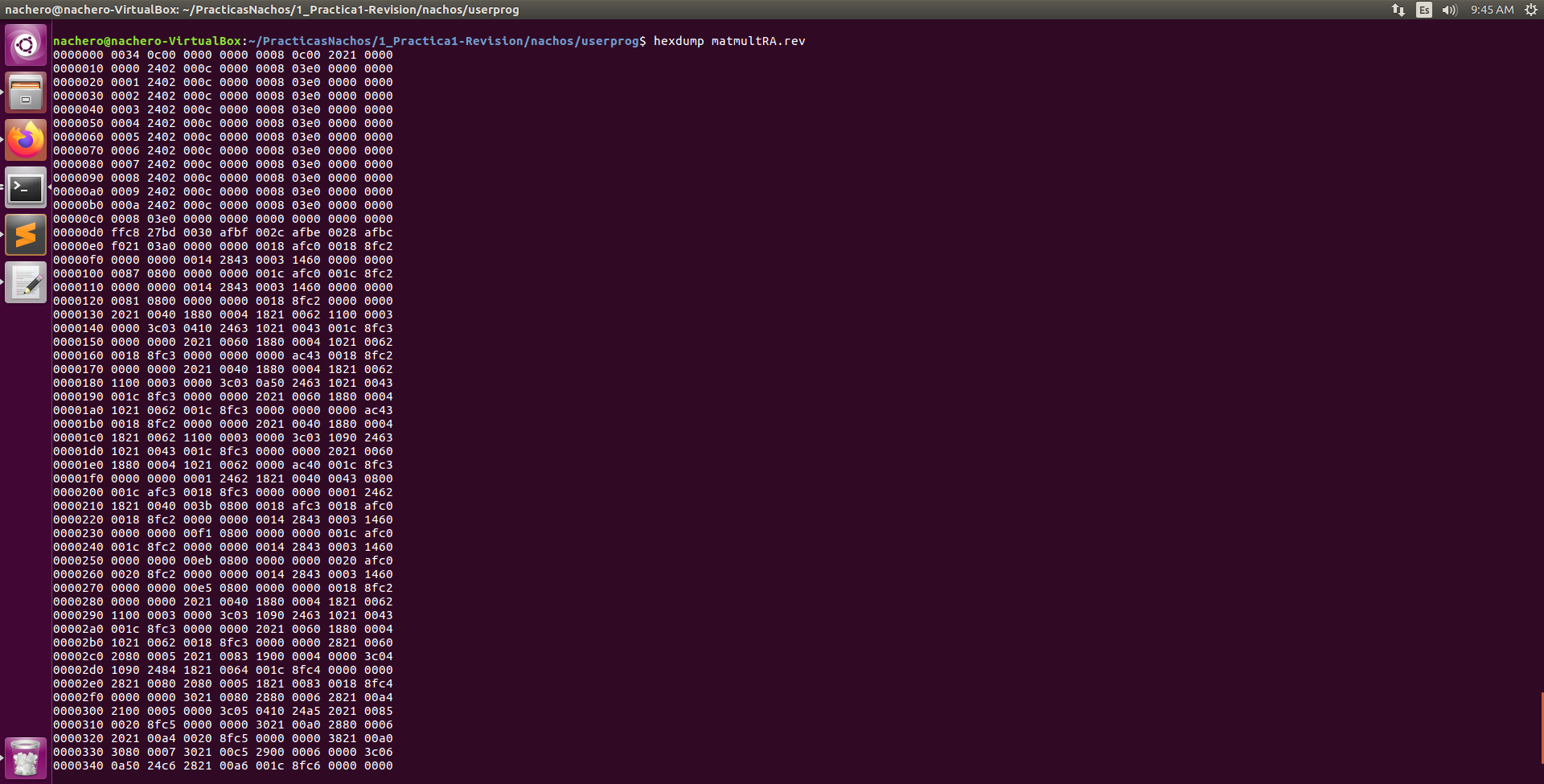
**h)** **Despliegue el contenido del archivo *haltRA.rev* usando el comando hexdump**

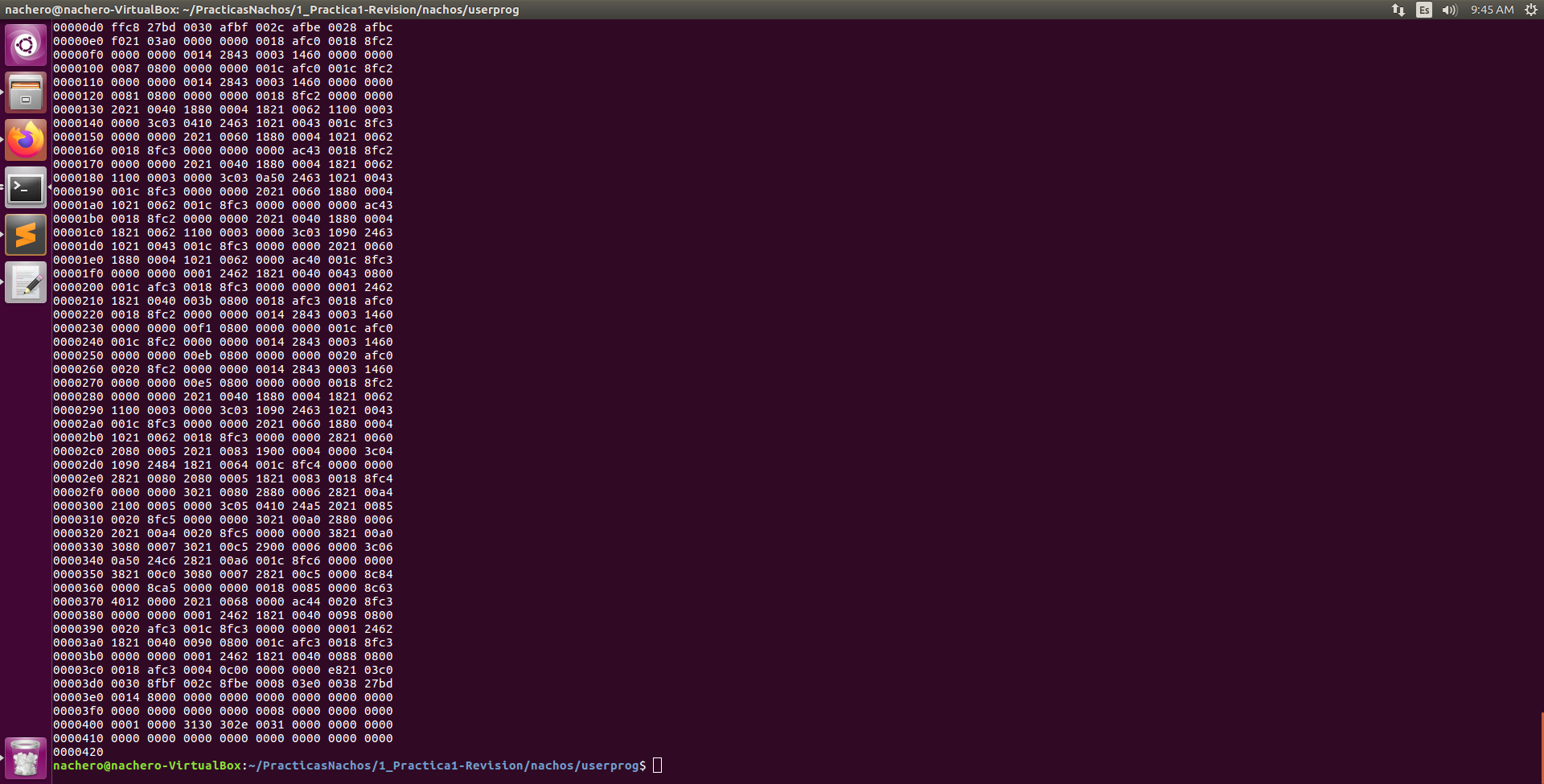
***Anexe una captura de pantalla***

****

**i)** **Despliegue el contenido del archivo *matmultRA.rev* usando el comando hexdump**

***Anexe una captura de pantalla***

****

****