**Estructura y contenido**

1. **Definición y especificación de requerimientos:**
   1. **Definición general del proyecto de software:**  
      **Idea general:**   
      Desarrollar en lenguaje ensamblador un programa que reciba la ruta de un archivo como parámetro, que lea dicho archivo y lo procese, de tal manera que se muestre el contenido del archivo en formato hexadecimal y en código ASCII. La salida será similar a la producida por el comando **hexdump –C**.  
        
      **Objetivos:**   
      Se brinda un mecanismo mediante el cual se logran procesar archivos enviados y detallar su información de una manera específica  
        
      **Usuarios:**   
      Este programa está dirigido hacia aquellos usuarios que busquen obtener una manera de observar el contenido de un determinado archivo, tanto en notación hexadecimal como ASCII.  
      Para poder utilizarlo, los usuarios deberán tener orientación en la utilización de la consola de comandos.  
      El presente informa está dirigido hacia aquellos usuario y/o desarrolladores interesados en obtener información respecto a cómo fue desarrollado este sistema.
   2. **Especificación de requerimientos del proyecto:**
      * **Requisitos generales:**   
        El proyecto consta de las siguientes pautas:
        1. Se debe tomar el contenido de un archivo y mostrarlo por pantalla de la siguiente forma:  
           **[Dirección base] [Contenido hexadecimal] [Contenido ASCII]**
        2. En caso de que se esté intentado imprimir un carácter no imprimible, éste será reemplazado por un “.”.
      * **Requisitos funcionales:**   
        Para que el programa se ejecute correctamente, se deben seguir las siguientes pautas  
        Para que la expresión aritmética sea correctamente evaluada, ésta debe cumplir con ciertas pautas:
        1. La invocación al programa se deberá realizar de la siguiente manera:  
           $ ./volcar [-h] <archivo>  
           Donde:
           + <archivo>: Denota la ruta del archivo a ser leído.
           + -h: Hace referencia al parámetro de ayuda. Este parámetro es opcional
        2. Si se ingresan 3 parámetros, se asumirá que el segundo parámetro hace referencia al parámetro –h (ayuda) y se ignorará el tercer parámetro. En caso de que se haya ingresado otro parámetro como primer parámetro, el sistema terminará la ejecución.
        3. Si se ingresan 2 parámetros, se asumirá que éste hace referencia a la ruta del archivo. En caso de que no sea una ruta válida, el sistema terminará la ejecución.
        4. Si se ingresa una cantidad de parámetros diferentes a 2 o 3, el sistema terminará le ejecución por error de parámetros.
        5. El formato del archivo debe hacer referencia al directorio completo de la carpeta contenedora del archivo y al final el archivo mismo.  
           Ej.: /home/alumno/Escritorio/archivo.txt
      * **Información de autoría y Legacy del proyecto:**   
        Este proyecto forma parte de un desarrollo original. Es compatible bajo el sistema operativo Debian, a partir de la versión gcc (Debian 4.9.2-10)4.9.2.
      * **Alcances del sistema:**  
        Este programa realiza su tarea aplicada a archivo de tamaño máximo 1MB. En caso de que el archivo enviado tengo un tamaño mayor a 1MB, el programa procederá a imprimir la información de dicho archivo hasta el byte que complete la cantidad de 1MB, los bytes siguientes a este serán descartados y no se mostraran por pantalla.
   3. **Procedimientos de instalación y prueba:**  
      ***Procedimientos de desarrollo*:**
      * **Herramientas utilizadas:**   
        Este sistema fue desarrollado mediante LXTerminal, es decir, fue desarrollado a través de una consola de comandos, en el sistema operativo Debian, en la versión gcc (Debian 4.9.2-10)4.9.2.
      * **Planificación:**   
        Se comenzó el programa desarrollando una función mediante la cual verificar los parámetros ingresados por el usuario. Para el caso de que estos parámetros fueran correctos se procedió a desarrollar los procedimientos que se realizarían dependiendo de dichos parámetros, y en caso de que fuesen incorrectos, a comunicar esto y terminar la ejecución del programa.  
        En caso de que solicitase información respecto al sistema, se implementó un método con el cual brindar al usuario un sistema de ayuda del programa.  
        En cuestión a que se hubiera recibido el archivo a ser leído, se continuó con crear un método a través del cual mostrar la información del archivo según lo establecido en las pautas del proyecto, indicando dirección base, contenido hexadecimal y contenido ASCII.  
        Para poder ejecutar la función relacionada al archivo, se inició con buscar la forma de lograr abrir dicho archivo y obtener la información en él contenida. Una vez obtenida ésta información, se siguió con encontrar una forma para lograr mostrar el contenido de acuerdo a las pautas ya mencionadas.

**Procedimientos de instalación y prueba:**

* + - **Requisitos no funcionales:**   
      El correcto funcionamiento del sistema se verá afectado si no se tiene en cuenta el máximo tamaño permitido del archivo a leer, lo cual está especificado en la sección “Alcances del sistema”. En caso de que esto no se cumpla, el archivo será leído, pero no completamente, sino que se leerá hasta el máximo de contenido ya establecido
    - **Obtención e instalación**:Para lograr obtener este sistema alcanza con colocar en una misma carpeta todos aquellos archivos de extensión .asm
    - **Especificaciones de prueba y ejecución:** Este sistema fue desarrollado en el sistema operativo Debian, versión gcc (Debian 4.9.2-10)4.9.2. El sistema funciona correctamente bajo esta plataforma. No obstante, esto no implica que bajo otros sistemas operativos este programa no se ejecute correctamente, sino que no ha sido testeado bajo otras plataformas.  
      Para poder ejecutar éste programa, se debe realizar el siguiente procedimiento:
      1. Ejecutar la consola de comandos del sistema operativo.
      2. Posicionarse en la carpeta contenedora del archivo perteneciente al programa (volcar.asm).
      3. Ejecutar el comando “ld -o volcar volcar.o”, con el fin de enlazar el código objeto.
      4. Ejecutar el comando “/.volcar [-h] <archivo>”. La explicación de estos parámetros se encuentra establecida en la sección de requisitos funcionales.

1. **Arquitectura del sistema**

* **Descripción jerárquica:**   
  Este sistema fue desarrollado con una estructura monolítica, ya que consta de un módulo compuesto por diferentes rutinas entrelazadas de tal forma que cada una pueda llamar a cualquier otra en caso de que la necesite.
* **Diagrama de módulos:**   
  Este proyecto fue desarrollado en un único módulo denominado “volcar”

volcar.asm

* **Descripción individual de los módulos:**   
  *Módulo de volcar:*
  + **Descripción general y propósito:**   
    Es un módulo que consiste en una secuencia de funciones que se reiteran en bucle para lograr mostrar por pantalla el contenido del archivo enviado como parámetro de una forma específica, para este caso, mostrar en filas de 16 bytes el contenido de tal forma que se visualice  
    [Dirección base] [Contenido hexadecimal] [Contenido ASCII]
  + **Responsabilidad y restricciones:**   
    Este módulo lleva a cabo cada una de las responsabilidades del sistema, por lo cual se encarga de:
    - Procesar los parámetros ingresados y verificar si son correctos, mostrando un mensaje en caso de que no lo sean
    - Abrir el archivo en la ruta especificada, mostrando un mensaje de error correspondiente en caso de que la ruta ingresada sea correcta.
    - Mostrar un mensaje de ayuda en caso de que se solicite (a través del parámetro –h).
    - Procesar la información del archivo, mostrándola de manera que se represente [Dirección base] [Contenido hexadecimal] [Contenido ASCII]
  + **Dependencias:**   
    Al tratarse del módulo único del sistema, éste no tiene dependencias de otros módulos, ya que los medios necesarios para que se ejecute se encuentran dentro de él mismo.  
    Sin embargo, depende fuertemente de funciones de ejecución del sistema, es decir, de que se ejecuten las llamadas al sistema para llevar a cabo diversas funciones del módulo.
* **Dependencias externas:**   
  En cuanto a dependencias externas, se necesitan de llamadas al sistema. Las llamadas realizadas en éste programa son:  
  sys­\_write:  
  Necesario para lograr por pantalla el contenido del archivo ingresado.  
  stdout:  
  Para indicarle al sistema que se está produciendo una salida estándar.  
  sys\_open:  
  Para llevar a cabo la apertura del archivo.  
  sys\_read:  
  Se lo necesita para lograr leer el contenido del archivo.  
  sys\_exit:  
  Para indicar la finalización del programa.  
   **Este sistema fue desarrollado utilizando el ensamblador Yasm, sobre arquitectura Intel x86, haciendo uso de las llamadas al sistema provistas por el sistema operativo Linux. La versión del sistema operativo usada es gcc (Debian 4.9.2-10)4.9.2.**

1. **Descripción de procesos y servicios ofrecidos por el sistema**Este sistema ofrece un único servicio, cuya funcionalidad radica en procesar archivo de cualquier formato (binario, imagen, texto u otro), de tamaño máximo 1MB.  
   La ruta debe especificar tanto el nombre del archivo (con su extensión) como el directorio completo de la carpeta contenedora del archivo. Por ejemplo: /home/alumno/Escritorio/archivo.txt

Una vez ingresado la ruta del archivo, el programa se encargara de abrir el archivo en caso de que la dirección sea válida, y obtener la información en él contenida.

Luego, ira leyendo cada uno de los bytes contenidos en el archivo e irá imprimiéndolo según las pautas del proyecto, con lo cual se realizan lecturas de a 16 bytes para cada impresión por pantalla. Si se llega al máximo de bytes posibles a ser leídos (1MB), los siguiente bytes del archivo serán descartados.  
Para realizar estas tareas, el programa cuenta con los siguientes datos:

Datos de entrada:  
Recibe como parámetro un archivo de cualquier formato (binario, imagen, texto u otro). En caso de que la ruta del archivo no sea correcta, se muestra por salida estándar un mensaje indicando lo sucedido.  
Si la dirección de archivo es correcta, se procede a capturar la información contenida en el archivo y procesarla con los diversos datos internos del sistema.  
Los datos internos utilizados son:  
  
Datos internos:  
Buff: Almcena el contenido del archivo  
BUFFLEN: Tamaño de buff  
  
\_LineaMemoria: Almacena la siguiente dirección base a imprimir  
\_LongLineaMemoria: Longitud de LineaMemoria  
\_LineaHexa: Almacena la información de los siguientes 16 bytes a imprimir en formato hexadecimal  
\_LongLineHexa: Longitud de LineaHexa  
\_LineaAscii: Contiene la información de los siguiente 16 bytes a imprimir en formato ASCII.  
\_LongLineaAscii: Contiene la longitud de LineaAscii  
FullLong: Almacena la longitud de LineaHexa  
TablaDigitosHexa: Es un string que contiene cada uno de los dígitos de base hexadecimal.  
archivo: String que almacena la dirección del archivo a leer  
manejadorArchivo: Toma el valor númerico retornado por el manejador de archivos luego de ejecutarse la instrucción de apertura del archivo.  
  
*Los siguientes datos son también datos internos, pero utilizados al producirse la salida estándar del sistema (salida por pantalla).*  
mensajeErrorParametros: Contiene un string indicando que se ingresaron mal los parámetros.  
longitudParametros: Entero que hace referencia al tamaño de mensajeErrorParametros.  
mensajeErrArchivo: Contiene un string que avisa que se ingresó un directorio incorrecto.  
longitudErrArchivo: Entero que indica el tamaño de mensajeErrArchivo.  
mensajeAyuda: String que muestra la ayuda disponible al ingresar el parámetro de ayuda.  
longitud: Entero que indica el tamaño de mensajeAyuda.

**5. Documentación técnica – Especificación API**   
*Métodos de volcar.asm:*

Itoa: Convierte un entero decimal a hexadecimal y devuelve su valor en string (buffer de chars). Recibe el entero en EAX y el buffer en EBX y retorna la cantidad de caracteres del string en EAX y el resultado de la conversión en el buffer. No modifica ningún otro valor de los registros con los que se llame a la función.

LimpiarLinea: Restaura “LineaHexa” llamando 16 veces a la rutina AgregarValorHex con parámetro 0. No tiene parámetros de entrada ni de salida y no modifica los valores de ningún registro.

AgregarValorHexa: Inserta un valor deseado tanto en la LineaHexa como en la LineaAscii. Recibe en EAX el valor que se desea insertar en las líneas y en EDX la posición de las mismas donde se desea insertar. Los valores a insertar los obtiene de “TablaAscii” y de “TablaDigitosHex” moviéndose sobre ellas. Solo modifica el valor del registro EAX.

ImprimirLinea: Impreme una línea entera en consola, esto es, la dirección base, la LineaHexa y la LineaAscii. Recibe en EDI el valor decimal de la dirección base a imprimir y devuelve en EDI el valor de la dirección base de la próxima línea. No modifica ningún otro registro.

leerFile: Llena el buffer Buff con caracteres del archivo. Puede quedar el buffer lleno o no depende la cantidad de caracteres que queden en el archivo. Retorna la cantidad de bytes leidos en EBP y retorna el valor 0 en ECX ya que luego se usara de puntero a las posiciones del buffer.

\_start: A partir de esta rutina inicia el programa. Se evalúa la cantidad de parámetros de entrada, si son 2 deben ser el nombre del programa y el nombre/dirección del archivo y salta a la rutina “dosParametros”, si son 3 deben ser si o si el nombre del programa, la ayuda y el nombre/dirección del archivo y salta a la rutina “tresParametros” (ya que el formato de llamada al programa es ./volcar [-h] <archivo> si o si deben haber mínimo 2 parametros y en caso de haber 3, debe estar el opcional –h y aún así también el archivo ya que este parámetro no es opcional). Si hay menos de 2 parametros o mas de 3, salta a la rutina “errParametros”

dosParametros: Analiza los dos parámetros sabiendo que el primero es el nombre del programa y el segundo el archivo. Abre el archivo en modo lectura, si esto produce error, salta a “errArchivo” de lo contrario llama a “leerFile” para inicializar el primer buffer. Modifica esi y edi poniéndolos a ambos en 0 para usar de contadores.

AvanzarBuff: Avanza sobre el buffer Buff convirtiendo sus valores a hexadecimales, llamando a AgregarValorHex con el valor que se desea agregar a la línea y la posición y analizando si hay que imprimir una línea nueva o no. Usa ECX para avanzar por las posiciones del buffer. Si llegó al final del buffer, se terminó de procesar el archivo y se llama a la rutina “Listo”

ayuda: Esta rutina solo se encarga de llamar al kernel con syscall write para imprimir por consola el mensaje de ayuda “mensajeAyuda” y luego terminar el programa con terminación normal.

errParametros: Esta rutina solo se encarga de llamar al kernel con syscall write para imprimir por consola el mensaje “mensajeErrorParametros” y luego terminar el programa con terminación anormal.

errArchivo: Esta rutina solo se encarga de llamar al kernel con syscall write para imprimir por consola el mensaje “mensajeErrArchivo” y luego terminar el programa con terminación anormal por error en el archivo de entrada.

Listo: Solo llama a la rutina ImprimirLinea para imprimir la última línea y luego continua con la rutina “salir”.

salir: Esta rutina solo se encarga de llamar al kernel con syscall exit y código de terminación normal.