Proyecto de arquitectura
Data-set SGY

Nombre: Moises Merza CI:25547115
Nombre: Harold Franco CI:2833094

May 2023

1 ¿Que es un archivo SGY?

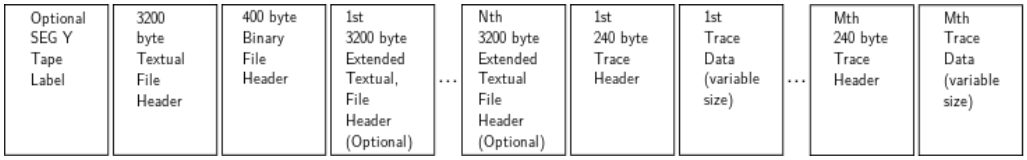
Un archivo SGY es utilizado en la industria de la exploración sísmica para almacenar datos sísmicos. Estos archivos contienen información recopilada por equipos sísmicos, que se utilizan para analizar las capas subterráneas de la Tierra y detectar la presencia de petróleo, gas u otros recursos naturales.

También contienen datos sísmicos que se han recopilado utilizando técnicas de adquisición sísmica, como la reflexión sísmica, que implica enviar ondas de sonido a través del subsuelo y medir los tiempos en que las ondas reflejadas regresan a la superficie. Estos datos se almacenan en un formato de archivo binario especializado que permite su procesamiento y análisis posterior.

Es utilizan comúnmente en la industria del petróleo y el gas para la exploración y producción de hidrocarburos. Los datos sísmicos contenidos en los archivos SGY se utilizan para crear imágenes de las capas subterráneas de la Tierra y para identificar posibles depósitos de petróleo y gas.

2 ¿Como esta compuesta el data-set SGY?

El data set esta compuesto por las siguientes secciones Textual Header, Binary Header y la traza, se le muestra una imagen relacionada a como se divide la información del dataset SGY.



- **Textual Header**: Al leer los primeros 3200 bits en el archivo se encontrar información adicional sobre los datos sísmicos almacenados en el archivo. A diferencia del Binary Header (que se hablara mas adelante), que contiene información en un formato binario especializado, el Textual Header almacena información en un formato de texto legible.

  Puede incluir información sobre el proyecto de adquisición sísmica, el equipo utilizado para recopilar los datos, los parámetros de procesamiento y análisis de los datos, la ubicación geográfica de la adquisición sísmica y otros metadatos relevantes. Esta información puede ser útil para entender el contexto en el que se recopilaron los datos sísmicos y para asegurarse de que se procesen y analicen correctamente.

- **Binary Header**: Luego de leer los 3200 bits anteriores del Textual Header ahora se leen 400 bits del Binary Header contiene información sobre la configuración del equipo de adquisición sísmica utilizado para recopilar los datos, como el número de canales, la frecuencia de muestreo, la duración de la adquisición y el tamaño de cada traza. También puede incluir información sobre la ubicación geográfica de la adquisición sísmica, la profundidad de la capa terrestre en la que se realizó la medición y otros metadatos relevantes.

  La información contenida en el Binary Header es crítica para la correcta interpretación de los datos sísmicos contenidos en el archivo SGY. Los programas de procesamiento y análisis de datos sísmicos utilizan esta información para interpretar los datos y crear imágenes de las capas geológicas subterráneas.

- **Traza**: Ya leidos El Textual Header y Binary Header que fueron 3600 bits en total, se leeran en adelante 240 bits, donde se almacena La informacion de la traza.

  las trazas son los datos sísmicos que se han recopilado en un punto de adquisición específico. Cada traza representa una medición de la reflexión de una onda sísmica en un punto determinado debajo de la superficie terrestre.

3 Algoritmo del programa

Realizamos un programa en C++ que pueda leer un dataset SGY, Colocamos dicho algoritmo en nuestro repositorio en Github, explicare aquí como leemos el data set y las funciones que usamos en nuestro programa.

El Programa inicia leyendo los 3 datasets en modo lectura, Colocamos un menú para que el usuario elija cual de los 3 desea abrir, este menú continua ejecutándose hasta que el usuario desee salir del menú lo cual finaliza el programa.

Luego de que el usuario elija un archivo del menú empieza el programa, en nuestro algoritmo usamos la librería nmap esta librería nos ayuda a leer los dataset SGY, y usamos la función "nmap()" esto nos ayuda a mapear el archivo SGY, es decir, asignamos un espacio en memoria virtual para leer el dataset.

Explicaremos como funciona la funcion "nmap()" Usaremos los parametros de nuestro algoritmo para explicar cada parametro que es lo que hace

mmap(NULL, sb.stsize, PROTREAD, MAPPRIVATE, fd, 0);

Esta línea de código se está utilizando la función mmap para asignar una región de memoria en el espacio de direcciones virtuales del proceso.

Los parámetros que se están utilizando son:

- **NULL**:Este parámetro indica que el sistema operativo debe elegir la dirección base de la región de memoria a asignar. Alternativamente, se podría especificar una dirección base específica.
- **sb.stsize**: Este parámetro indica el tamaño de la región de memoria que se quiere asignar.
- **PROTREAD**: Este parámetro especifica los permisos de protección de la región de memoria. En este caso, se está indicando que la región de memoria sólo será utilizada para lectura.
- **MAPPRIVATE**: Este parámetro indica que la región de memoria será accesible únicamente por el proceso actual. Cualquier cambio que se haga en la región de memoria no será visible en otras copias del mismo archivo.
- **fd**:Este parámetro indica el descriptor de archivo que se obtuvo
- **0**: Este parámetro indica el desplazamiento dentro del archivo donde comenzará a mapearse la región de memoria. En este caso, se está comenzando a mapear desde el inicio del archivo.

Ya explicado como es la función "nmap()" seguimos con la explicación.

Luego de mapear el dataset se establece una condición si cubre lo suficiente el espacio en memoria virtual ya que el dataset puede ser tan grande que puede llegar al límite de la memoria virtual.

Al mapear dicho dataset se empieza a escanear el encabezado, esta información se encuentra al principio del dataset que son 3200 bits, para leerlo usamos una función llamada "memcpy()", Se utiliza para copiar una secuencia de

bytes desde una dirección de memoria origen(lo asignamos antes con la función "nmap()") a una dirección de memoria destino (en este caso leer 3200bits). Ya leído el encabezado usamos unas de las librerías para convertir esa información que está en binario a texto legible.

Explicaremos como funciona la funcion "memcpy()" Usaremos los parametros de nuestro algoritmo para explicar cada parametro que es lo que hace

memcpy(cabecera, mem,3200);

Los parámetros que se están sutilizando son:

- **Cabecera**: Es un puntero que apunta a la dirección de memoria de destino donde se copiará la secuencia de bytes.
- **mem**:Es un puntero que apunta a la dirección de memoria de origen desde donde se copiará la secuencia de bytes.
- **3200**:Es el número de bytes que se copiarán desde la dirección de memoria de origen a la dirección de memoria de destino. En este caso, se están copiando los primeros 3200 bytes de mem a cabecera.

La llamada a memcpy en este caso está copiando los primeros 3200 bytes de la región de memoria apuntada por mem a la región de memoria apuntada por cabecera.

Ya explicado como es la función "memcpy()" seguimos con la explicación.

Ahora hacemos el mismo proceso para leer la Informacion binaria, ya sabiendo que se leyeron 3200bits del encabezado se empezara desde ese punto de origen a leer en adelane 400 bits que representan la información binaria, todo esto con dicha función "memcpy()", ya analizado la información pasa a convertirse a texto legible con ayuda de las librerías. Al tener la información binaria podemos extraer el número de trazas que contiene el dataset, por lo que usaremos esa información.

Ya lo que sigue es leer las trazas, luego de leer el encabezado, la información binaria y tener la cantidad de trazas que contiene el dataset, se procederá hacer un bucle que partirá desde el origen de memoria sumando los 3200bits del encabezado y también 400 bits de la información binaria hasta el número total de trazas y se leerán 240 bits. Las trazas se leen en 240 bits en adelante, por lo que al leer 240 bits se sumara al origen de memoria, así hasta llegar al total de trazas.

Mientras corra el bucle se irán imprimiendo por consola las trazas con sus respectivas coordenadas y se generaran dos archivos.txt que son: Encabezado.txt e InformacionBinaria.txt, del dataset que se haya elegido.

4 Casos bases

Estos casos bases se probaron en 2 computadoras lo llamaremos compuadora A compuadora B.

Componentedes de la Computadora A:

- Sistema operativo: Ubuntu 22
- Disco duro: SDD 120GB
- Memoria RAM: 8GB

Componentedes de la Computadora B:

- Sistema operativo: Windows 10
- Disco duro: HDD 500GB
- Memoria RAM: 8GB

Nos basamos en 3 casos bases pequeño(24kb), intermedio(380mb) y grande (1,3Gb), explicaremos si hubo relevancia en cada caso en las 2 computadoras.

- **Caso pequeño (24kb)**: El programa corre perfectamente en ambas computadoras, generando 2 archivos.txt Encabezado.txt (Text Header) e InformacionBinaria.txt(Binary Header), y en la terminal se muestra por pantalla las trazas con sus coordenadas que son en total 61 trazas y concuerda con la InformacionBinaria.txt. El tiempo de ejecucion al correr el programa en la computadora A fue de 0,004s mientras que que la computadora B fue de 1.4s un gran diferencia.
- **Caso intermedio (380mb)**: El programa compila igual de bien como el caso pequeño, generando los 2 archivos.txt, Encabezado.txt e InformacionBinaria.txt, mostrando en la terminal una cantidad de 312 trazas y concuerda con la InformaciónBinaria.txt. El tiempo de ejecucion al correr el programa en la computadora A fue de 0,019s mientras que que la computadora B fue de 2.8s sigue siendo una gran diferencia de tiempo.
- **Caso grande (1,3GB)**: Compilando este caso grande no hay errores pero si un cambio en los archivos.txt generados este cambio paso en ambas computadoras, el Encabezado.txt no muestra la información debida y esto es porrque viene dividido en partes la información del Encabezado por lo grande que es, pero como nos enfocamos en las trazas lo dejaremos así, sin embargo el archivo InformaciónBinaria.txt si da la información debida y tambien muestra por pantalla todas las trazas que son 13.728 coordenadas y concuerda con la InformaciónBinaria.txt. El tiempo de ejecucion al correr el programa en la computadora A fue de 0,3s mientras que que la computadora B fue de 4.5s viendo como iba el patron de los casos anteriores se sabia que la computadora A sigue siendo mejor.

Teniendo en cuenta los datos obtenidos en como compilaban el programa la computadora A y computadora B gana por velocidad la computadora A esto por su componente del Disco duro que teine una SSD en vez de una HDD que es el caso de la computadora B, recordemos que Un disco duro SSD es generalmente mejor que un disco duro HDD en términos de velocidad. Los discos duros SSD no tienen partes móviles y almacenan datos en chips de memoria flash, lo que les permite acceder a los datos de manera más rápida que los discos duros HDD, que utilizan un brazo mecánico para leer y escribir datos en discos giratorios.