Instituto Tecnológico de Costa Rica

Escuela de Computación

Curso: Principios de Sistemas Operativos

Segundo Proyecto

El objetivo de este proyecto consiste en programar un empacador de archivos. Este es el tipo de funcionalidad que provee el comando *tar* en ambientes UNIX.

El programa *tar*, es usado para almacenar archivos y directorios en un solo archivo. Dentro de los entornos Unix *tar* aparece como un comando que puede ser ejecutada desde la línea de comandos de una consola de texto o desde un simple terminal. El formato del comando *tar* es, comúnmente

```
tar <opciones> <archivoSalida> <archivo1> <archivo2> ... <archivoN>
```

donde *archivoSalida* es el archivo resultado y *archivo1*, *archivo2*, etc., son los diferentes archivos que serán "empaquetados" en *archivoSalida*.

Las *opciones* más comunes son las siguientes:

- -c, –create : crea un nuevo archivo
- -x, –extract : extraer de un archivo
- -t, –list: listar los contenidos de un archivo
- –delete: borrar desde un archivo
- -u, –update: actualiza el contenido del archivo
- -v, –verbose: ver un reporte de las acciones a medida que se van realizando
- -f, –file: empaquetar contenidos de archivo, si no está presente asume la entrada estándar.
- -r, –append: agrega contenido a un archivo

Ejemplos

1. Si queremos empacar un directorio llamado "html" y guardar los datos en "html-paq.tar", lo haríamos con la instrucción

   ```
   tar -cvf html-paq.tar html
   ```

2. Si queremos desempaquetar todo el contenido de un archivo llamado xxx.tar podemos utilizar un comando como este

   ```
   tar -xvf xxx.tar
   ```

3. Para archivar el contenido de tres archivos doc1.txt, doc2.txt y data.dat

   ```
   tar -cvf foo.tar doc1.txt doc2.txt data.dat
   ```

4. Si ahora se desea eliminar el contenido del archivo data.dat se ejecutaría

```
tar --delete -vf foo.tar data.dat
```

5. Para agregar ahora un nuevo archivo test.doc a foo.tar se ejecutaría

```
tar -rvf foo.tar test.doc
```

Los contenidos se desempaquetarán en el mismo directorio donde estamos situados.

Implementación

Se deberá programar el comando star ("simple tar", NO "estrella"), de tal forma que acepte los comandos básicos mostrados anteriormente. Note que debe permitir tanto el empacar archivos individuales como directorios completos en forma recursiva. Para desarrollar su programa usted debe tomar en cuenta los siguientes aspectos:

- El archivo "tar" deberá estar estructurado mediante bloques de espacio continuo que serán asignados en forma individual a los archivos que se deben almacenar.
- Al crear un archivo empacado éste se crea del tamaño necesario para almacenar los archivos agregados. Cuando se borra algún contenido, el archivo empacado no cambia de tamaño sino que se lleva registro de los espacios (huecos) liberados. Si posteriormente se agrega nuevo contenido entonces se reutiliza el espacio libre. Si aún así el nuevo contenido no cabe, se hace crecer el archivo empacado.
- En el archivo *tar* se almacenará una tabla que indique la dirección en donde empieza cada archivo y su tamaño. Internamente también se debe llevar registro del contenido de los directorios. Para ello se deberá utilizar espacio adicional ien el archivo empacado. El directorio se almacenará como cualquier archivo pero internamente contará con secuencia de nombres de archivo y sus posiciones en el archivo "tar". Puede asumir que los nombres de archivo no son mayores a los 256 caracteres.
- Para asignar el espacio a cada archivo se debe buscar un *hueco* en el archivo *tar* en el cual se pueda introducir. En este caso se utilizará la estrategia de "el primer ajuste" para realizar esta asignación. Para ello se debe llevar un control interno de los *huecos* sin utilizar en el archivo y se debe realizar la *fusión* de dichos huecos cuando sea necesario.
- Si no hay un hueco del tamaño adecuado en el archivo se debe utilizar alguno de los siguientes mecanismos: (a) expandir el tamaño del archivo ó (b) realizar la compactación del archivo para eliminar huecos pequeños (opcional)
- Tome en cuenta que un archivo que se agrega puede ya existir en el archivo empacado. Es decir, lo que se desea hacer es actualizar su contenido. Para

esto existe la opción update (-u) que sobrescribirá el contenido de un archivo siempre y cuando su fecha de modificación sea posterior a la del archivo empacado.

- Debe programar una solución eficiente, es decir, debe minimizar la cantidad de espacio utilizado tanto por los archivos como por las estructuras internas.
- Este programa no utilizará los derechos de acceso, que normalmente almacenaría un archivo empacado *tar* en ambiente UNIX.
- No utilice ningún archivo adicional (ni siquiera archivos temporales o intermedios) para implantar este programa, con excepción del archivo *.tar*.

Pruebas de correctitud

Para realizar las pruebas de correctitud se incluirá un nueva opción -d, –dump que imprimirá en pantalla la tabla de contenido del archivo, así como la lista de bloques libres. La idea es poder seguir el rastro de las diferentes operaciones conforme se van ejecutando en el archivo.

Luego se deberán ejecutar diversos conjuntos de pruebas a su programa con diferentes combinaciones de operaciones de: crear, eliminar, actualizar y desfragmentar contenido. Estas pruebas deben ser realizadas tanto con archivos individuales como con directorios completos.

Utilizado las salidas de la opción *dump* se deberán documentar los resultados de dichas pruebas y se deberá mostrar el correcto funcionamiento de las diferentes operaciones.

Documentación

Se deberá generar una documentación formal, en formato pdf, en donde se describan las diferentes etapas del desarrollo del proyecto, las decisiones de diseño que se tomaron, los mecanismos de programación utilizados, y los resultados de las diferentes pruebas al programa.

Dicha documentación deberá incluir al menos las siguientes secciones:

- Introducción
- Descripción del problema (este enunciado)
- Definición de estructuras de datos
- Descripción detallada y explicación de los componentes principales del programa:
  - Mecanismo de acceso a archivos y directorios
  - Estructura de la tabla de asignación de espacio
  - Estrategia de administración de bloques libres
  - Procedimiento de desfragmentación del archivo (opcional)

- Análisis de resultados de pruebas
- Conclusiones sobre rendimiento

Puede agregar diagramas a la documentación que le ayuden a ilustrar las diferentes estructuras de datos y algoritmos principales.

Forma de evaluación

- Programa star (80 %)
  - Operaciones sobre archivos: crear, extraer y listar (20%)
  - Operaciones sobre archivos: borrar, actualizar y agregar (20%)
  - Operaciones sobre directorios: crear, extraer y listar (20%)
  - Operaciones sobre directorios: borrar, actualizar y agregar (20%)
  - Procedimiento de desfragmentación del archivo (10%)
- Análisis de pruebas (20%)
  - Pruebas de correctitud: comando dump (10%)
  - Análisis de resultados y conclusiones (10%)

Consideraciones generales

- El proyecto debe ser elaborados en grupos de a lo más dos personas. Se puede desarrollar en forma individual pero bajo ninguna circunstancia se aceptarán grupos de tres o más personas.
- No se permite la copia de código entre grupos de estudiantes, utilizar código tomado desde Internet, como tampoco es permitido utilizar clases o librerías adicionales (desarrolladas por terceros) para simplificar el manejo de: lectura de archivos, lectura de directorios, manejo de bloques en disco, empacar datos, desfragmentación, o cualquier otra función del programa.
- El proyecto debe ser resuelto mediante lenguaje C y ambiente Unix.
- Se deberá crear un archivo zip que incluya: el código fuente de todos los programas desarrollados, binarios generados, y documentación en formato pdf.
- Puede buscar en Internet mayor información sobre el uso de las diferentes opciones del comando *tar*, sin embargo, note que existen muchas otras opciones que no serán implementadas en este proyecto.