TEC | Tecnológico de Costa Rica

Proyecto 2

Black & White Filesystem

Octubre, 2022

Objetivo

Realizar una re-implementación de algunas de las funciones de un filesystem en el espacio de usuario del Sistema Operativo GNU/Linux.

Datos Generales

- **Fecha de Entrega:** Jueves 24 de Noviembre de 2022 antes de las 23:59:59 GMT-6.
- **Fecha de Revisión:** Viernes 25 de Noviembre de 2022.
- **Lenguaje:** C o Rust para GNU/Linux
- **Recurso Humano:** Grupos de 2 o 3
- **Valor de la asignación:** 20 %

Profesor

Kevin Moraga
kmoragas@ic-itcr.ac.cr
Escuela de Computación

Introducción

BWFS es un sistema de archivos que reside en el espacio de usuario. Este sistema de archivos tiene como objetivo utilizar los imágenes blanco y negro para almacenar archivos. Para BWFS el espacio físico donde se almacena su información y toda su estructura se encuentra en pixels de color blanco y negro, definidos en la creación del FS.

Requerimientos Funcionales

BWFS

Se realizará la re-implementación de las siguientes funciones utilizando la biblioteca FUSE:

- getattr
- create
- open
- read
- write
- rename
- mkdir
- readdir
- opendir
- rmdir
- statfs
- fsync
- access
- unlink
- flush
- lseek

mkfs.bwfs

Este binario consiste en la creación de un nuevo sistema de archivos tipo BWFS. Teniendo en cuenta lo siguiente:

- Cada bloque de BWFS podrá tener un máximo de 1000 px por 1000 px.
- Se le solicitará al usuario introducir un nuevo "passphrase". Este se almacenará, y se utilizará como firma para encontrar el inicio del FS. (opcional)
- Toda la información relevante a la organización del FS (ej. Superblock, o el manejo de espacio libre) deberá ser cifrada a partir de la contraseña del usuario. El resto del FS permanecerá sin cifrar. (opcional)
- El sistema de archivos deberá de utilizar i-nodos como estructura de indexación de bloques. (opcional, puede utilizar otra estructura)
- El sistema de archivos deberá permitir una estructura jerárquica de directorios. (opcional, puede ser solo 1 nivel)
- La sintaxis será:

```
mkfs.bwfs folder/
```

fsck.bwfs

Realiza un chequeo de consistencia de BWFS.

- La sintaxis será:

```
fsck.bwfs folder/
```

mount.bwfs

Se encarga de montar el BWFS en algún punto de montaje perteneciente al FS del Sistema Operativo.

- Para el usuario es posible especificar cual archivo corresponde al inicio del FS o bien, el FS se encargará de encontrar la firma en los archivos BW que se encuentren en la carpeta a montar.
- La sintaxis será:

```
mount.bwfs folder/ mountpoint/
```

Requerimientos Técnicos Adicionales

- El desarrollo se debe de realizar utilizando el lenguaje de programación C.
- Es necesario utilizar la biblioteca FUSE para la implementación.
- El Sistema de Archivos debe ser persistente en disco.
- Será posible crear archivos de cualquier tipo y cualquier tamaño (siempre y cuando sea menor al tamaño definido en la creación).
- Será necesario definir una estrategia para administración la fragmentación del BWFS.

Extra

- Realizar la implementación de redimensión del FS. (15%)

Aspectos Administrativos

Entregables

- Código fuente del programa que cumpla los requerimientos funcionales y técnicos.
- Código fuente de mkfs.bwfs
- Código fuente de fsck.bwfs
- Código fuente de mount.bwfs
- Código fuente de BWFS.
- Binario del programa, compilado para una arquitectura x86.
- Fuente de la documentación en Markdown o en Latex y luego a PDF.
- PDF con la documentación.
- Un sistema de archivos impreso.

Kick-off

Se deberá realizar un documento para la clase siguiente, con el siguiente contenido:

1. **Introducción**: Debe presentar el problema, utilizando una redacción propia. Incluyendo un pequeño esquema definiendo la estrategia para solucionar el problema asignado.
2. **Ambiente de desarrollo**: Indicar las herramientas que se utilizarán durante la elaboración de la tarea. Incluyendo entorno de desarrollo, forma de debugging, flujo de trabajo en un sistema de control de versiones.
3. **Control de Versiones**: Se debe de incluir y compartir el enlace al repositorio que se utilizará en el control de versiones seleccionado.
4. **Diagrama UML**: diagrama de clases en UML, una explicación de las decisiones de diseño.

Documentación

Las siguientes son las instrucciones para la documentación. NO LA IMPRIMA. Además la documentación se debe de realizar utilizando MD con Latex.

1. **Introducción**: Presentar el problema. Puede "reciclar" partes del enunciado de la tarea programada.
2. **Ambiente de desarrollo**: Indicar las herramientas usadas para implementar la tarea.
3. **Estructuras de datos usadas y funciones**: Se debe describir las principales funciones y estructuras utilizadas en la elaboración de esta asignación.
4. **Instrucciones para ejecutar el programa**: Presentar las consultas concretas usadas para correr el programa para el problema planteado en el enunciado de la tarea y para los casos planteados al final de esta documentación.
5. **Actividades realizadas por estudiante**: Este es un resumen de las bitácoras de cada estudiante (estilo timesheet) en términos del tiempo invertido para una actividad específica que impactó directamente el desarrollo del trabajo, de manera breve (no más de una línea) se describe lo que se realizó, la cantidad de horas invertidas y la fecha en la que se realizó. Se deben sumar las horas invertidas por cada estudiante, sean conscientes a la hora de realizar esto el profesor determinará si los reportes están acordes al producto entregado.

6. **Autoevaluación**: Indicar el estado final en que quedó el programa, problemas encontrados y limitaciones adicionales. Adicionalmente debe de incluir el reporte de commits de git. Por otro lado, también debe incluir una calificación con la rúbrica de la sección "Evaluación" y "Autoevaluación" con cada ítem evaluado de 0 a 10.
7. **Lecciones Aprendidas** del proyecto: Orientados a un estudiante que curse el presente curso en un futuro.
8. **Bibliografía** utilizada en la elaboración de la presente asignación.
9. Es necesario documentar el código fuente.

Evaluación

- BWFS:
  - mkfs.bwfs: 14 %
  - fsck.bwfs: 5 %
  - mount.bwfs: 10 %
  - Funciones de la biblioteca: 26 %.
    - getattr
    - create
    - open
    - read
    - write
    - rename
    - mkdir
    - readdir (opcional)
    - opendir (opcional)
    - rmdir (opcional)
    - statfs
    - fsync
    - access
    - unlink
    - flush
    - lseek
- Documentación: 20 %
- Persistencia en Disco: 25 %
- Manejo de Fragmentación: 5 % (opcional)
- Otros opcionales: 5 % (la sumatoria total)

Autoevaluación

Realice una autoevaluación utilizando una escala [*1*] *Muy Malo,* [*2*] *Malo,* [*3*] *Regular,* [*4*] *Bueno y* [*5*] *Muy bueno*, de los siguientes rubros:

[1] [2] [3] [3] [5]	Aprendizaje de mkfs.
[1] [2] [3] [3] [5]	Aprendizaje de fsck.
[1] [2] [3] [3] [5]	Aprendizaje de mount.
[1] [2] [3] [3] [5]	Aprendizaje de implementacion de funciones.
[1] [2] [3] [3] [5]	Aprendizaje de Diseño de Filesystem.

Aspectos Adicionales

Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones:

1. Sí no se entrega documentación, automáticamente se obtiene una nota de 0.
2. Sí el código no compila se obtendrá una nota de 0, por lo cuál se recomienda realizar la defensa con un código funcional.
3. El código debe ser desarrollado en el lenguaje especificado en los Datos Generales, en caso contrario se obtendrá una nota de 0.
4. Sí no se siguen las reglas del formato del envío a través de Google Classroom se obtendrá una nota de 0.
5. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto.
6. Cada grupo tendrá como máximo 20 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cuál se recomienda tener todo listo antes de ingresar a la defensa.
7. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final de la presente asignación.
8. Cada grupo es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberá avisar al menos 2 días antes de la revisión al profesor para coordinar el préstamo de estos.
9. Durante la revisión podrán participar asistentes, otros profesores y el coordinador del área.
10. Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.