Laboratorio 4: Permisos

Version original: 2014 Rafael Carrascosa

Introduccion

En este laboratorio se implementaran permisos para el sistema de archivos. Los permisos sirven para regular las operaciones que se pueden realizar sobre archivos y directorios, pudiendo así prevenir daños accidentales o intencionales a los usuarios del SO.

El esquema típico de permisos de UNIX considera 3 tipos de permisos para archivos y directorios:

- Lectura (R).
- Escritura (W).
- Ejecución (X).

Además, cada archivo o directorio guarda permisos RWX distintos para 3 clases de usuarios:

- El dueño del archivo/directorio (U).
- El grupo al cual pertenece el archivo/directorio (G).
- Cualquier otro usuario (O).

Es decir, cada archivo o directorio en un sistema de archivos UNIX tiene (al menos) 9 bits de permisos.

Para este laboratorio implementaremos una simplificación de este esquema en la cual cada archivo del sistema de archivos de XV6 tiene una sola tripla de permisos RWX que aplican a cualquier usuario que acceda al archivo (de todas formas XV6 es mono-usuario).

Enunciado

Los permisos que deberán implementar son sobre archivos (opcionalmente directorios) y son los siguentes:

- Lectura (R): Los datos del archivo pueden ser leidos si y solo si esta presente.
- Escritura (W): Se pueden escribir datos en el archivo si y solo si esta presente.
- Ejecución (X): Se puede ejecutar el archivo si y solo si esta presente.

Para lograrlo deberán:

- Añadir permisos a los inodos.
- Implementar una syscall y un programa de usuario **`chmod`**.
- Hacer cumplir los permisos en las operaciones de archivo.
- Escribir un informe.

Añadir permisos a inodos

Primero deberán estudiar el sistema de archivos de `xv6`. Comenzar por `fs.c` y prestar atención a las diferentes capas de abstracción: bloques, inodos, archivos.

La estructura de inodos en disco del FS de xv6 no tiene espacio libre para poner permisos, así que

vamos a tener que robarle 3 bits a algún otro campo del inodo. La sugerencia de la cátedra es robarle 3 bits al campo `minor`. Para robar estos bits deberán investigar y aprender sobre "C bit fields".

El `sizeof` de las `structs` que modifiquen **debera ser el mismo** antes y despues de sus cambios.

Deberán prestar atención a que los permisos están correctamente sincronizados entre la cache en memoria y el disco.

Implementar `chmod`

Este laboratorio requiere implementar una única syscall: **`chmod`**. Esta syscall permite cambiar los permisos de un archivo. La signatura de la syscall (del lado de usuario) debería ser:

```
int chmod(char *filename, int perms);
```

Dónde la `filename` es el path al archivo al cual se quiere cambiar sus permisos y `perms` es un argumento con flags que determina los permisos que quedarán habilitados en este archivo.

Los bits de `perms` válidos son: `S_IREAD`, `S_IWRITE`, `S_IEXEC` (tomar inspiración de esto) y deberán definirlos ustedes en algún `.h` del kernel que consideren apropiado.

Además de implementar la syscall, deberán implementar un programa de espacio de usuario análogo al programa `chmod` de UNIX (ver manpage). La sintaxis de uso deberá ser:

```
chmod [perms] [filename]
```

Dónde `perms` es un numero entero que códifica los permisos, por ejemplo un número del 0 al 7 y `filename` es el path hasta el archivo al cual se desea cambiar sus permisos.

Hacer cumplir los permisos

Luego deberán modificar donde sea pertinente para implementar las restricciones de los permisos de lectura, escritura y ejecución.

El más simple de implementar es el de ejecución, así que la cátedra recomienda comenzar por ahí y probar las siguientes acciones:

```
ls  # deberia funcionar
chmod 6 ls  # asumiendo que 6 es R/W pero no X
ls  # deberia fallar
# Salir de xv6 y volver a entrar
ls  # deberia fallar de nuevo
chmod 1 ls  # asumiento que 1 es X pero no R/W
ls  # deberia funcionar
```

Para implementar permisos de lectura y escritura se recomienda ver las variables `readable` y `writable`.

Si hay dudas con respecto a la semántica de los permisos, estos deberán comportarse de igual forma que los de un UNIX (p.ej. Ubuntu).

Una vez implementados los permisos deberán ejecutar el programa `testchmod` provisto por la catedra para ayudarlos a encontrar errores. Si el programa falla usted tiene un error. No vale la contrarrecíproca, **no es cierto** que "si el programa no falla usted no tiene errores". **No insista**, no es cierto y no es válido como argumento.

Escribir un informe

Finalmente deberán escribir un breve informe dónde indiquen las decisiones más importantes que tomaron junto a sus justificaciones.

Extras

- Implementar permisos para directorios.
- Modificar `mkfs` para que cuando copie los archivos del sistema host para crear la imagen de disco de xv6 preserve los permisos de usuario que estan presentes en el FS de la máquina host(la máquina host es típicamente la PC de laboratorio o su laptop).

Consideraciones y entrega

Siendo este el último laboratorio de la materia se espera que los alumnos sean capaces de encontrar lo que necesiten dentro de `xv6` por su propia cuenta así como se espera mayor **autonomía** a la hora de buscar información.

- Deberán entregar via commits+push a su grupo en bitbucket.
- Deberán crear un repositorio git nuevo `Lab4` dentro del grupo `so2014gXX` con un directorio `xv6` dentro sobre el cual deberán hacer sus modificaciones. No copiar el de otro laboratorio, comenzar en limpio.
- El *coding style* deberá respetar a rajatabla las convenciones de `xv6`.