Sistemas Operativos 5º semestre. Grado II

Primer Parcial. Sistema de Ficheros. Noviembre de 2016.

Ejercicio 1 (1 punto)

Explique qué es la tabla intermedia del servidor de ficheros del sistema operativo.

- **a)** ¿Qué información contiene? Describa sus campos.
- **b)** ¿Qué permite esta tabla que no podría hacerse sin ella? Indique su relación con otras tablas.
- **c)** ¿Qué llamada al sistema ocupa una nueva entrada en esta tabla?
- **d)** ¿En qué circunstancias se libera una entrada de esta tabla?

Solución

a) La tabla intermedia del servidor de ficheros es una tabla única del sistema operativo que permite relacionar los descriptores de fichero a través de los cuales los procesos interactúan con los ficheros abiertos, y que se almacenan en el BCP del proceso, con la copia en memoria de los i-nodos que están en la tabla de i-nodos. Cada descriptor de fichero de cada proceso tiene asociado un valor entero que representa la fila dentro de esta tabla intermedia a través de la cual tiene que manipular el i-nodo del fichero al que este descriptor hace referencia.

   Esta tabla contiene los siguientes campos:

   - Nº i-nodo: Identifica al fichero que se encuentra abierto.
   - Posición: Puntero de lectura/escritura compartido por todos los procesos que tengan descriptores duplicados apuntando a esta misma fila.
   - Referencias (o ndups): Número de descriptores (duplicados) de fichero que apuntan a esta misma fila.
   - Modo de apertura con el que fue abierto el fichero (R,W, RW).

b) Esta tabla permite que varios descriptores (que podrían corresponder, potencialmente, a varios procesos distintos) compartan un único puntero de lectura/escritura sobre un fichero abierto. Cada entrada de esta tabla intermedia es apuntada por uno o varios descriptores de fichero de uno o varios procesos y apunta a su vez a una única entrada de la tabla de i-nodos.

c) Las llamadas open() y creat() añaden una nueva entrada en esta tabla, con el puntero de lectura/escritura apuntando al comienzo del fichero y el número de referencias igual a 1.

d) Una entrada de esta tabla se considera liberada cuando el número de referencias es igual a 0. Esto quiere decir que los procesos que tenían descriptores de ficheros apuntando a esta entrada de la tabla los han ido cerrando (close()) y se libera al cerranse el último.

Ejercicio 2 (2 puntos)

Implemente en C y para Unix el mandato: `copyswap input output` que, considerando el archivo `input` como dos mitades, copia su contenido en el archivo `output` pero intercambiando dichas dos mitades. (Ej. `input` "abcd12345", `output` "12345abcd"). Cuando el archivo `input` sea de tamaño impar, considere la primera mitad un byte menor que la segunda. Considere que los archivos nunca tendrán un tamaño mayor de 10 MiB.

Solución


```
fd1 = open(argv[1], O_RDONLY);
```

```
            fdo = creat(argv[2], 0666);

            size = read(fdi, buff, SIZE);
            half = size/2;
            write(fdo, &buff[size], size-half);
            write(fdo, &buff[0], half);

            close(fdi);
            close(fdo);
            return 0;
}
```

Ejercicio 3 (3 puntos)

Sea la siguiente visión parcial del contenido del sistema de ficheros de un sistema:

Num	T	U	G	O	User	Group	Ruta
1	d	rwx	r-x	r-x	root	root	/
2	d	rwx	rwx	rwt	root	root	/tmp/
3	-	r--	-w-	--x	yoda	jedi	/tmp/salida
4	d	rwx	r-x	r-x	root	root	/home/
5	d	rwx	r-x	--x	yoda	jedi	/home/yoda/
6	-	rwx	r-s	--x	yoda	jedi	/home/yoda/cp
7	d	rwx	r-x	--x	r2d2	robot	/home/r2d2/
8	-	r--	-w-	--x	r2d2	robot	/home/r2d2/entrada
9	l	rwx	rwx	rwx	r2d2	robot	/home/r2d2/temporal -> /tmp/

Donde: Num es la linea de esta tabla; T es el tipo de objeto; U, G y O son los grupos de permisos (*user*, *group* y *other*); y la notación A -> B indica que el enlace simbólico A apunta a la ruta B.

Considere que el usuario r2d2 (del grupo robot) invoca, desde su *home*, el mandato de copia:

```
/home/yoda/cp    ./entrada    temporal/salida
```

Conteste las siguientes preguntas detallando cómo el sistema operativo decodifica la ruta indicada, mostrando: número de linea, grupo de permisos y permiso concreto, que se comprueba a cada paso.

a) ¿Podría ejecutarse el mandato indicado? ¿Qué identidad tendría el proceso resultante?

b) ¿Podría el proceso abrir el archivo de entrada? Detalle cada paso de la decodificación.

c) ¿Podría el proceso abrir el archivo de salida? Detalle cada paso de la decodificación.

Solución

a) El proceso con identidad r2d2:robot realiza la llamada exec de la ruta /home/yoda/cp, cuya decodificación sigue los siguientes pasos:

[1, O, x] para cruzar el directorio raiz / y no somos ni el propietario ni de su grupo.

[4, O, x] para cruzar el directorio /home/ y no somos ni el propietario ni de su grupo.

[5, O, x] para cruzar el directorio /home/yoda/ y no somos ni propietario ni del grupo.

[6, O, x] para ejecutar el mandato /home/yoda/cp y no somos ni propietario ni grupo.

[6, G, s] como se ha conseguido ejecutar y el ejecutable tiene activo el bit SETGID, el proceso pasará a tener como identidad efectiva de grupo la del propietario del archivo ejecutable, esto es, jedi.

b) El proceso con identidad efectiva r2d2:jedi realiza la llamada open para lectura de la ruta

[9, -, -] es un enlace simbólico, no se comprueban permisos, se sigue la redirección `/tmp/`.

[1, O, x] para cruzar el directorio raiz `/` y no somos ni el propietario ni de su grupo.

[2, O, x] para cruzar el directorio `/tmp/` y no somos ni el propietario ni de su grupo. Disponemos de este permiso aunque el que se muestra es el Stiky bit (`t`).

[2, O, t] el Stiky bit se usa en directorios de uso público para que los archivos sólo puedan ser borados (`unlink`) por sus respectivos propietarios.

[3, G, w] tras la decodificación del enlace, seguimos decodificando lo que nos quedaba de la ruta, esto es, `salida` desde el punto `/tmp/` por donde ibamos. No somos el propietario, pero (por efecto del bit SETGID) nuestro grupo efectivo (`jedi`) es el del propietario. El permiso validado es de escritura pues estamos abriendo para escritura.

Ejercicio 4 (4 puntos)

En un sistema Unix con tamaño de bloque 1KiB, con un disco con sector de 512B y dos particiones (a las que hay que dar formato de sistema de ficheros tipo Unix) de las características siguientes:

Partición	Tamaño	Tipo de SF	Tamaño medio de fichero	Punto de montaje
`/dev/sda1`	100 GiB	UFS	6144 B aprox.	`/`
`/dev/sda2`	512 GiB	UFS	0.5 MiB aprox.	`/home/`

a) ¿Qué tamaño de agrupación escogería para cada sistema de ficheros? Razone su respuesta.

b) ¿Cuánto espacio de disco ocuparían los datos de un archivo de 1KiB, en cada caso?

Para el caso de `/dev/sda2` y usando el tamaño de agrupación que usted haya escogido y haciendo las suposiciones que considere razonables, explique:

c) ¿Cuántos punteros a agrupación cabrán en una agrupación de indirección?

d) ¿Cuánto ocupará el mapa de bits de inodos en este sistema? Indique cálculos y unidades.

Solución

a) Dados los tamaños medios de fichero de ambas particiones escogeremos como tamaño de agrupación aquél que se acerque más a dichos tamaños medios y que se corresponda con 2^n bloques.

En el caso de /dev/sda1, el tamaño de agrupación sería 4KiB u 8KiB.
En el caso de /dev/sda2, el tamaño de agrupación sería exactamente 0.5MiB dado que es una potencia de dos: 2^{19} bytes.

b) Los datos ocuparán siempre un número entero de agrupaciones del sistema de ficheros, pues es la unidad de asignación de espacio a los ficheros y no se puede asignar media agrupación. Por lo tanto, en /dev/sda1 el archivo ocuparía 4KiB u 8KiB y en /dev/sda2 512KiB.

c) Es fácil comprobar que 32 bits son suficientes para direccionar el máximo número de agrupaciones posibles para este tamaño de partición. Para ello, lo que haremos será dividir el tamaño de la partición entre el tamaño de la agrupación. Esto nos dará el número de agrupaciones que tenemos en este sistema de ficheros:

$$\frac{512\ GiB}{\quad} = \frac{2^{39}\ B}{\quad} = 2^{20}$$

$$\frac{512\ KiB}{4\ B} = \frac{2^{19}\ B}{2^2\ B} = 2^{17}$$

Por lo tanto, en una agrupación de indirección cabrán 2^{17} punteros a agrupación.

d) En realidad, esta cuestión la hemos respondido antes al calcular el máximo número de agrupaciones que puede haber en la partición (que hemos visto que es 2^{20}). Teniendo en cuenta esto, podemos considerar que el máximo número de ficheros coincide con el máximo número de agrupaciones, es decir, 2^{20}. Por lo tanto, necesitaremos un bit por cada uno de estos ficheros para marcar el estado del i-nodo (libre/ocupado), es decir, 2^{20} bits o, lo que es lo mismo, 1Mi bits, o lo que es igual 128KiB.
