Auxiliar 7 – CC41B

Prof.: Luis Mateu Aux.: Juan Manuel Barrios

9 de noviembre de 2007

Año 2002, Examen, Pregunta 2

Parte a.

El tamaño típico de un bloque de archivo en Unix es 1KB mientras que en Windows es de 4 KB. Señale la principal ventaja del tamaño de bloque usado por Unix y la principal ventaja del tamaño de bloque de Windows.

Parte b.

Se tiene un archivo Unix de 1GB. El programa de la izquierda lee bloques al azar en el archivo. Para ello utiliza la función `drand48`, que entrega números aleatorios entre 0 y 1, y la llamada al sistema `lseek`, que señala al sistema de archivos a partir de qué posición se debe realizar la próxima lectura. El programa de la derecha lee la misma cantidad de bloques pero secuencialmente.

```
char buff[1024];                    |   char buff[1024];
int i, fd=open(....);               |   int i, fd=open(...);
for(i=0; i<1024*1024; i++){         |   for(i=0; i<1024*1024; i++)
    int bloq=drand48()*1024*1024;   |       read(fd, buff, 1024);
    int pos=bloq*1024;              |
    lseek(fd, pos, SEEK_SET);       |
    read(fd, buff, 1024);           |
}                                   |
```

El disco se publicita con un tiempo de acceso de 10 milisegundos y una tasa de transferencia de 20MB/seg. Estime el tiempo de ejecución de ambos programas. Haga supuestos razonables y explique como realiza sus cálculos.

Parte c.

Se tiene un archivo Unix en una partición que usa bloques de 512 bytes. Haga el diagrama con la descomposición en inodo, bloques de indirección (simples y dobles) y bloques de datos del archivo suponiendo que su largo es:

```
(12 + 128) * 512 + 1 bytes
```

Solución

Parte a.

- En Unix la principal ventaja es que se desperdicia menos espacio para archivos pequeños.
- En Windows, la ventaja es que un bloque más grande significa menor fragmentación y se recorre con menos saltos el archivo completo.

Parte b.

En el programa de la izquierda en cada iteración hay que pagar el costo de seek más el costo de transferencia. En el programa de la derecha habrá que pagar el costo de seek una sola vez al principio (para comenzar a leer el archivo) y luego en cada ciclo será necesario pagar sólo el costo de transferencia.

Supuesto 1: Los datos del archivo están contiguos en el disco.
Supuesto 2: No se da información del filesystem por tanto no podemos conocer el número de bloques de indirección que será necesario leer, pero si los bloques de indirección son mantenidos en caché será el mismo costo para los dos programas.

T_{seek} = Tiempo de hacer seek.
T_{1k} = Tiempo de transferencia de 1024 bytes.
T_{1g} = Tiempo de transferencia de 1024 * 1024 * 1024 bytes.

Trandom (tiempo programa izquierda):

$$1024 * 1024 * (T_{seek} + T_{1k}) = 1024 * 1024 * T_{seek} + T_{1g}$$

Tsecuencial (tiempo programa derecha):

$$T_{seek} + 1024 * 1024 * T_{1k} = T_{seek} + T_{1g}$$

Vemos que la diferencia de tiempo entre los dos programas lo impondrá el tiempo de costo de hacer seek.

Del enunciado tenemos que el tiempo de acceso es de 10 milisegundos y la tasa de transferencia es de 20MB/seg.

- T_{seek} = 0,01 seg.
- T_{1g} = 51,2 seg.
- Trandom = 10.537 segundos ~ 175 minutos.
- Tsecuencial = 51,21 segundos.

Si buscamos información de discos duros que se vendan actualmente en tiendas locales, para verificar como influiría en ellos esta diferencia en tiempo, encontramos lo siguiente:

Un disco SATA publica tiempo de acceso 8 ms y tasa de transferencia 300 MB/seg.
- Trandom = 140 minutos.
- Tsecuencial = 3,42 segundos

Un disco SCSI publica tiempo de acceso 4 ms y tasa de transferencia 320 MB/seg.
- Trandom = 70 minutos.
- Tsecuencial = 3,20 segundos

Parte c.

Si el tamaño bloque es de 512 bytes y la arquitectura es de 32 bits, entonces en cada bloque de indirección se pueden guardar 128 punteros (512 bytes / 4 bytes).

- Utilizando primero los 12 punteros de datos que hay en el inodo podemos referenciar: 12 * 512 bytes.
- Con el puntero de indirección simple podemos referenciar: 128 * 512 bytes.
- Luego será necesario utilizar el puntero de indirección doble para referenciar 1 byte.