Universidade de Brasília

Departamento de Ciência da Computação

Lista de Exercícios 2
Organização de Arquivos

Autores:

Giovanni M Guidini 16/0122660

Vitor F Dullens 16/0148260

Brasília

8 de Abril de 2018

Lista de Exercicios 2 **Organização de Arquivos**

Alunos: Giovanni Guidini 16/0122660; Vitor Dullens 16/0148260. CIC 116327
Prof. Oscar Gaidos Data: 29/03/2018

Exercicio 1

A - Descubra qual o tamanho da cluster do seu HD

B - Faça um programa que gere cinco arquivos contendo os mesmos dados, cada um com um fator de blocagem correspondente aos valores abaixo: 512 bytes, 1/4, 1/2, 3/4 e uma cluster.

C - O tamanho do registro e da cluser devem poder ser passados via linha de comando.

Resposta

A - O cluster do HD testado possui 4096 bytes, contendo 8 setores de 512 bytes cada.

Código

Obs.: Os arquivos produzidos contém NULL chars, por causa da diferença nos tamanhos dos blocos. Por causa disso não puderam ser adicionados neste pdf, mas colocamos as saídas do comando `stat` para cada arquivo, que indica a diferença no tamanho deles.

B - O código para esta questão está no apêndice A

Saídas:

Saídas

```
[gguidini@SilverAce testes]$ stat out_512.txt
  File: out_512.txt
  Size: 1024            Blocks: 8          IO Block: 4096   regular file
Device: 804h/2052d      Inode: 25037261    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/gguidini)   Gid: ( 1000/gguidini)
Access: 2018-04-06 21:41:03.400026397 -0300
Modify: 2018-04-06 21:41:03.400026397 -0300
Change: 2018-04-06 21:41:03.400026397 -0300
 Birth: -
```

Figura 1: Comando stat para o arquivo de 512 bytes. Note o campo `size`

```
[gguidini@SilverAce testes]$ stat out_1024.txt
  File: out_1024.txt
  Size: 1024            Blocks: 8          IO Block: 4096   regular file
Device: 804h/2052d      Inode: 25037253    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/gguidini)   Gid: ( 1000/gguidini)
Access: 2018-04-06 21:41:03.383359730 -0300
Modify: 2018-04-06 21:41:03.383359730 -0300
Change: 2018-04-06 21:41:03.383359730 -0300
 Birth: -
[gguidini@SilverAce testes]$ stat out_2048.txt
  File: out_2048.txt
  Size: 2048            Blocks: 8          IO Block: 4096   regular file
Device: 804h/2052d      Inode: 25037255    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/gguidini)   Gid: ( 1000/gguidini)
Access: 2018-04-06 21:41:03.386693063 -0300
Modify: 2018-04-06 21:41:03.386693063 -0300
Change: 2018-04-06 21:41:03.386693063 -0300
 Birth: -
[gguidini@SilverAce testes]$ stat out_3072.txt
  File: out_3072.txt
  Size: 3072            Blocks: 8          IO Block: 4096   regular file
Device: 804h/2052d      Inode: 25037257    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/gguidini)   Gid: ( 1000/gguidini)
Access: 2018-04-06 21:41:03.390026397 -0300
Modify: 2018-04-06 21:41:03.390026397 -0300
Change: 2018-04-06 21:41:03.390026397 -0300
 Birth: -
[gguidini@SilverAce testes]$ stat out_4096.txt
  File: out_4096.txt
  Size: 4096            Blocks: 8          IO Block: 4096   regular file
Device: 804h/2052d      Inode: 25037259    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/gguidini)   Gid: ( 1000/gguidini)
Access: 2018-04-06 21:41:03.393359730 -0300
Modify: 2018-04-06 21:41:03.393359730 -0300
Change: 2018-04-06 21:41:03.393359730 -0300
 Birth: -
```

Figura 2: Comando stat para arquivos de 1024, 3072 e 4096 bytes. Note o campo `size`

C - O código para esta questão está no apêndice B
Saídas:

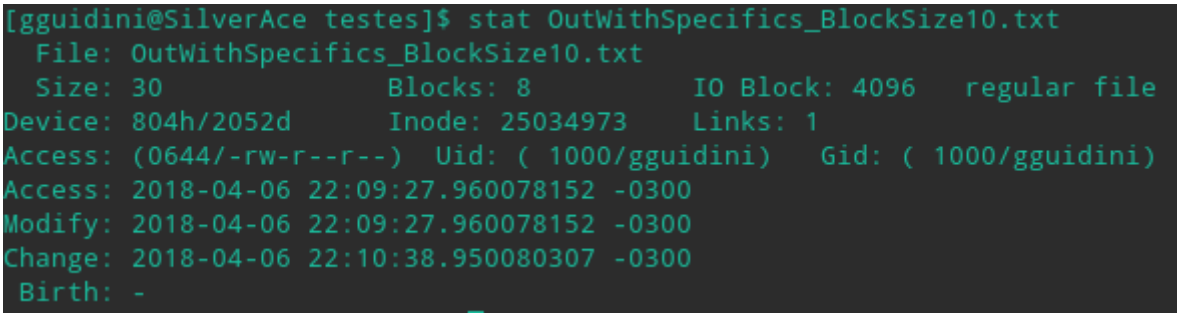

Figura 3: Comando stat para arquivo com bloco de 30 bytes. Note o campo `size`

Exercicio 2

Em uma fita de 2400 pés, densidade 30000 bpi, gap de tamanho 0,3 polegadas, ache o espaço necessário para armazenar 10000 registros de tamanho de 150 bytes.

Resposta

Feito o calculo utilizando a fórmula: S = n*(b+g) : onde 'S' é o espaço desejado, 'n' é o número de blocos (um registro por bloco), 'b' é o tamanho físico do bloco de dados e 'g' é o tamanho do gap.

Foi encontrado o valor: 3050 polegadas ou 254,17 pés ou 77,47 metros.

Exercicio 3

Segundo a visão do projetista de arquivo, determine:

A - O fator do bloco, com perdas mínimas, para armazenar registros de 128 bytes, em setores de 512 bytes, cujo bloco não pode ser superior ao cluster de 1536 bytes. Cada Página é igual a 4 blocos.

B - Faça a mesma coisa com registros de 125 bytes.

C - Qual é a fragmentação interna no caso 3a e no caso 3b?

Resposta

A - O melhor fator de blocagem seria de 12, que é o tamanho máximo permitido para um bloco (mesmo tamanho de um cluster), porque 128 * 12 = 1536 bytes, logo não haveria espaço desperdiçado.

B - O tamanho muito próximo dos registros não justifica mudança no fator de blocagem, que continua sendo ótimo em 12.

C - O caso 3a não tem fragmentação interna, pois os blocos comportam exatamente todos os registros. Já o caso 3b possui fragmentação de 36bytes por bloco utilizado.

Exercicio 4

Qual a vantagem de um arquivo em disco armazenado:

A - numa única extensão?

B - várias extensões distintas?

Resposta

A - O arquivo pode ser processado com o mínimo de tempo de busca, isto é, com um único seeking.

B - Para arquivos menores, dependendo da maneira que os arquivos são armazenados, o espaço alocado para eles pode ser muito maior do que o arquivo em si, causando fragmentação interna.

Exercicio 5

Quais são as vantagens e desvantagens da organização de trilhas em setores com grande capacidade.

Resposta

As vantagens estão principalmente no armazenamento de mais bytes por setor. Isso diminui a quantidade de setores que um arquivo ocupa, diminuindo o número de setores que precisam ser lidos para acessar o arquivo, o que aumenta a velocidade e eficiência do acesso aos arquivos. Se é sabido que os arquivos a serem utilizados são grandes é melhor aumentar o tamanho dos setores.

Se, por outro lado, os arquivos forem menores, a utilização de setores de grande capacidade vai implicar em espaços não utilizados, aumentando a fragmentação interna, o que é um problema.

A Exercício 1B

```cpp
// File for OA - UnB 2018/1
// By Giovanni M Guidini and Vitor F Dullens
// 1.b Make a program that generates five files with the same data,
// each with a different block factor of:
// 512B, 1/4, 1/2, 3/4 and 1 cluster.
// File used to collect data is dummy.txt and it's filled with lorem ipsu
#include <bits/stdc++.h>
#define CLUSTER 4096
using namespace std;

int main() {
    // opens original file
    fstream dummy ("test.txt", ios::in | ios::binary);
                                //    0.25       0.5          0.75          1
    vector<int> blockFactor = {512, CLUSTER/4, CLUSTER/2, 3*CLUSTER/4, CLUSTER};
    vector<char*> data[5]; // 5 vectors, one for each block factor.
    // get length of file:
    dummy.seekg (0, dummy.end);
    int length = dummy.tellg(), chunks;
    // reads file creating different sized blocks
    for(int i = 0; i < 5; i++){
        int bfr = blockFactor[i];
        dummy.seekg(0); // returns to begin of file
        chunks = 0;
        while(chunks < length){
            char* curr_blk = (char*) malloc(bfr);
            memset(curr_blk, 0, bfr);   // initialize with 0 if not all block
    will be filled

            if(chunks + bfr <= length)
                dummy.read(curr_blk, bfr);
            else
                dummy.read(curr_blk, length-chunks);
            data[i].push_back(curr_blk);    // adds data to that block factor
            chunks += bfr;
        }
    }
    printf("Blocks completed\n");
    printf("512K : %d blocks\n", (int) data[0].size());
    printf("CLUSTER/4 : %d blocks\n", (int) data[1].size());
    printf("CLUSTER/2 : %d blocks\n", (int) data[2].size());
    printf("3*CLUSTER/4 : %d blocks\n", (int) data[3].size());
    // creates file for output
    string filename = "";
    fstream out;
    for(int i = 0; i < 5; i++){
        int bfr = blockFactor[i];
        filename = "testes/out_" + to_string(bfr) + ".txt";
        out.open(filename, ios::out | ios::binary);
        // writed data from blocks to out file
        for(auto blk : data[i]){
            out.write(blk, bfr);
        }
        out.close();
    }
}
```

Listing 1: "Code for 1B - creating files with different block factors"

B Exercício 1C

```cpp
// File for OA - UnB 2018/1
// By Giovanni M Guidini and Vitor F Dullens

// 1.c Make a program that does what 1B does, but
// allowing register size and cluster size to be informed

// File used to collect data is dummy.txt and it's filled with lorem ipsu
#include <bits/stdc++.h>
using namespace std;

int main(int argv, char** argc) {
  // open test file
    fstream dummy ("test.txt", ios::in | ios::binary);
    if (!dummy.is_open()){
        printf("Didn't find file\n");
        return 1;
    }
    vector<char*> data;
    int regSize, cluSize;
    if(argv == 3){
        regSize = stoi(string(argc[1]));
        cluSize = stoi(string(argc[2]));
    }
    else{
        printf("Usage: ./1C <registerSize> <clusterSize>\n");
        return 1;
    }

    // Number of registers in the cluster
    int blk = cluSize/regSize;
    // block size
    int bfr = blk*regSize;
    // get length of file:
    dummy.seekg (0, dummy.end);
    int length = dummy.tellg(), chunks = 0;
    dummy.seekg(0); // returns to begin of file
    while(chunks < length){
            char* curr_blk = (char*) malloc(bfr);
            memset(curr_blk, 0, bfr);
            if(chunks+bfr <= length)
                dummy.read(curr_blk, bfr);
            else
                dummy.read(curr_blk, length-chunks);
            data.push_back(curr_blk);    // adds data to that block factor
            chunks += bfr;
        }
    printf("Reading finished. Total of %d blocks\n", (int) data.size());
    string filename = "OutWithSpecifics_BlockSize" + to_string(blk) + ".txt";
    fstream out (filename, ios::out | ios::binary);

    for(auto blk : data){
        out.write(blk, regSize);
    }
}
```

Listing 2: "Code for 1C - creating file with given register and cluster size"