Ficheiros

Programação I
2020.2021

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Reminder…

Sumário

Ficheiros de texto e ficheiros binários

Modos de acesso

Funções sobre ficheiros

Ficheiros de texto e binários

Ficheiros

Ficheiros de texto

- Informação guardada sob a forma de linhas de texto terminadas por um carácter especial
  - Linux, Unix → ‘\n’
  - Windows → ‘\r’ ‘\n’

Ficheiros binários

- Organizados em elementos de dimensão fixa
  - Os elementos podem ser de tipos simples ou estruturas
  - Não há necessidade de marcas de fim de elemento

Modos de acesso

Modo de acesso sequencial

- Elementos acedidos sequencialmente
  - Para aceder ao k-ésimo elemento é necessário aceder previamente aos elementos, 0, 1, ..., k-1

Modo de acesso direto (ou aleatório)

- Elementos acedidos por qualquer ordem
  - Os elementos têm de ser necessariamente de dimensão fixa

Ficheiros e modos de acesso

Ficheiro de texto

- Modo de acesso sequencial
  - Para alterar a k-ésima linha num ficheiro é necessário
    1. criar um novo ficheiro
    2. copiar as k-1 primeiras linhas
    3. escrever a nova linha
    4. copiar as restantes linhas

Ficheiro binário

- Modo de acesso sequencial
- Modo de acesso direto

Utilização de ficheiros

Ficheiro de texto

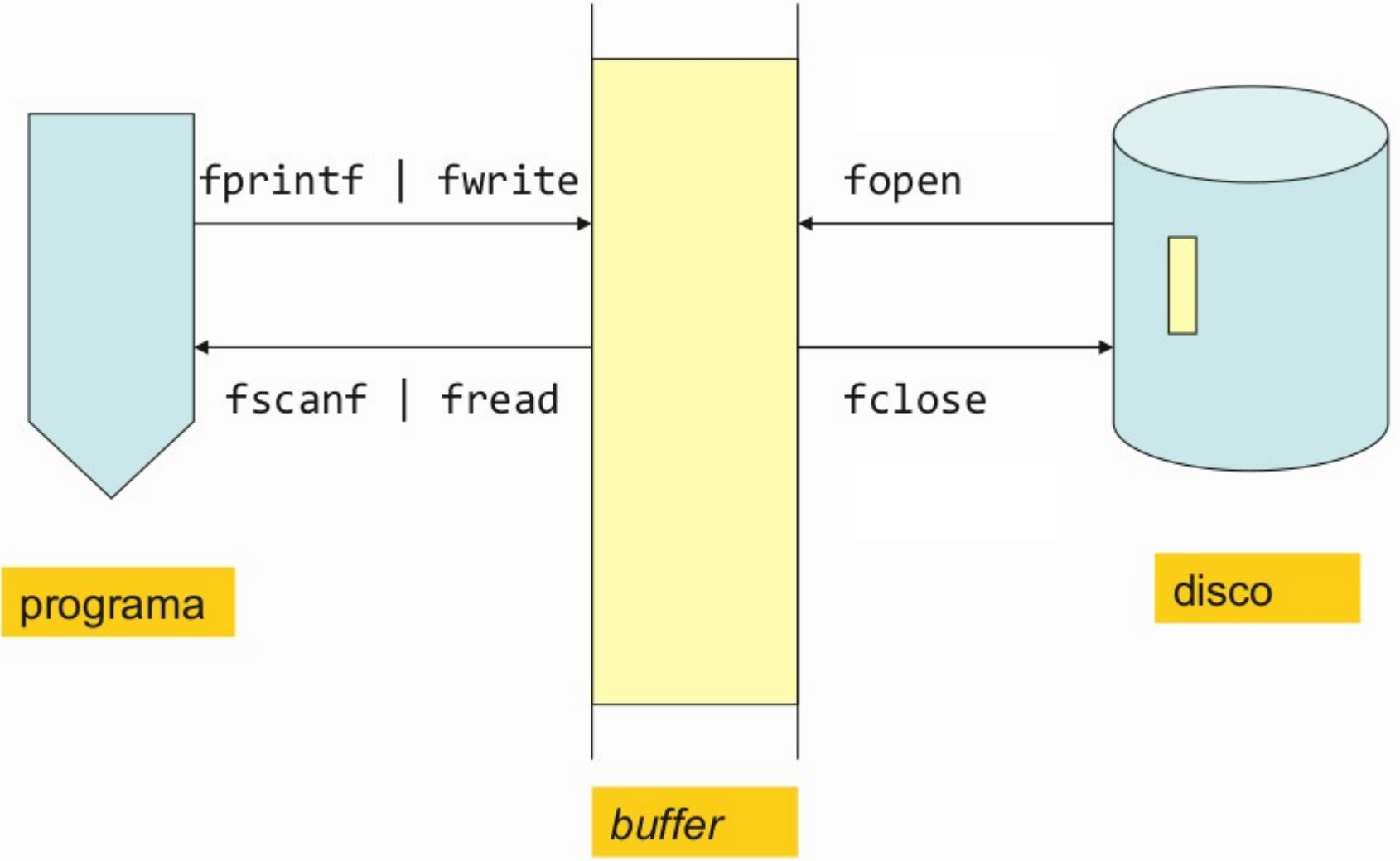
Confortável utilização pois podem ser editados e inspecionados com qualquer editor

Devem ser usados quando os dados armazenados sofrem poucas alterações e a quantidade é moderada

Ficheiro binário

Utilizados para grandes quantidades de dados que necessitem operação rápida de acesso diretoe que sofrem contínuas alterações

Operações


fprintf | fwrite
fscanf | fread
fopen
fclose
programa
buffer
disco

Funções

FILE

Estrutura para manipulação de ficheiros

Definida em `stdio.h`

fopen() e fclose()

```
FILE *fopen( char *filename, char *mode )
```

Abre o ficheiro com nome `filename`, usando o modo de acesso `mode` e associa-o uma estrutura FILE.

Modos de acesso

- “r” → leitura
- “w” → escrita
- “a” → acrescentar
- “b” → formato binário

Devolve **`NULL`** quando existe um erro na abertura de ficheiro

```
int fclose( FILE *f )
```

Fecha o ficheiro associado a f

Exemplo

```
FILE *f;

char nomef[100];
printf(“nome do ficheiro? “);
scanf(“%s”, nomef);
f = fopen(nome,”w”);
...
fclose(f);
```

fprintf() e fscanf()

Ficheiros de texto

```
int fprintf( FILE *f, string_de_formato
[, expr1, ..., exprn] )
```

Comportamento idêntico ao printf

```
int fscanf( FILE *f, string_de_formato, &var1
[,&var2 ..., &varn] )
```

Comportamento idêntico ao scanf

getc() e putc()

Ficheiros de texto

```
char getc( FILE *f )
```

Corresponde à função `getchar()`

```
void putc( char c, FILE *f )
```

Corresponde à função `putchar( char c )`

feof()

EOF

End Of File: Carácter indicador de fim de ficheiro

fgetc() devolve EOF após a leitura do último carácter do ficheiro

int feof(FILE *)

Função que testa o final de ficheiro

Devolve <> 0 se final ficheiro; 0, caso contrário

Exemplo

```
#include <stdio.h>
int main() {
   FILE *f = fopen("new.txt", "r");
   int c = getc(f);
   while (c != EOF) {
      putchar(c);
      c = getc(f);
   }
   if (feof(f))
      printf("\n Reached to the end of file.");
   else
      printf("\n Failure.");
   fclose(f);
   return 0;
}
```

Ler uma linha de um ficheiro

```
#include <stdio.h>
int fgetline(FILE *fp, char line[], int max){
    int nch = 0; int c;
    max = max - 1;          /* guardar espaço para '\0' */
    while((c = getc(fp)) != EOF || c!= ‘\n’)
        if(nch < max) {
           line[nch] = c;
           nch = nch + 1;
        }
    if(c == EOF && nch == 0) return EOF;
    line[nch] = '\0';
    return nch;
}
...
char line[MAXLINE];
...
fgetline(ifp, line, MAXLINE);
```

fread() e fwrite()

Ficheiros binários

```
int fread(void *pt, int size, int n, FILE *f)
```

Lê para pt n elementos de tamanho size

Devolve o nº de elementos corretamente lidos

```
int fwrite(void *pt, int size, int n, FILE *f)
```

Escreve de pt n elementos de tamanho size

Devolve o nº de elementos corretamente lidos

Exemplo

```
#include <stdio.h>

main(){
    int v[4] = {1200, 200, 56, 14};
    FILE *f;
    f = fopen("int.bin", "wb");
    fwrite(v, sizeof(int), 4, f);
    fclose(f);
    printf("Escrita do ficheiro terminada\n");
}
```

fseek()

```
int fseek( FILE *fich, long salto,int origem )
```

Função para acesso direto aos elementos. Ficheiros binários

`salto` → nº de bytes a "andar" para trás ou para a frente em relação à origem

`origem` → 3 valores possíveis:

- `SEEK_SET`: início do ficheiro;
- `SEEK_CUR`: posição corrente do ficheiro;
- `SEEK_END`: fim do ficheiro.

ftell() e rewind()

```
long ftell( FILE *fich )
```

Indica a posição corrente em bytes

```
void rewind( FILE *fich )
```

Volta ao início do ficheiro

Exemplo

```
#include <stdio.h>
#include <string.h>

main(){
  int n; FILE *f;
  f = fopen("real.dat", "wb");
  fseek(f, 0, SEEK_END); /* vai para o fim */
  n = ftell(f);
  printf("Bytes %d; Nums reais:%d\n", n, n/(sizeof(float)));
  fclose(f);
}
```

Exemplo – ficheiro de texto

```
#include <stdio.h>

main(){
  int i;
  char cidade[4][10] = {"Porto", "Aveiro", "Coimbra", "Lisboa"};
  float temp[4] = {12.5, 11.8, 12.2, 14};
  FILE *fout; /* declara variável de ficheiro */

  fout = fopen("exp.txt", "w"); /* abre ficheiro de texto para escrita */
  for( i=0; i<4; i++ )
    fprintf( fout,"%s %f\n", cidade[i], temp[i] );
  fclose( fout );
  printf( "Escrita do ficheiro terminada\n" );
}
```

Controlo de erros

```
#include <stdio.h>
main() {
  int i,k;
  char *cidade[4] = {"Porto", "Aveiro", "Coimbra", "Lisboa"};
  float temp[4] = {12.5, 11.8, 12.2, 14};
  FILE *fout;

  fout = fopen("exp.txt", "w");    /* abre ficheiro para escrita */
  if (fout == NULL) {   /* se apontador inválido */
    printf("Impossivel criar o ficheiro\n");
    exit(1);  /* termina com erro */
  }
  else {
    for (i = 0; i < 4; i++) {
      k = fprintf(fout,"%s %f\r\n", cidade[i], temp[i]);
      printf("%d\n", k);
    }
    fclose(fout);
    printf("Escrita do ficheiro terminada\n");
  }
}
```

stderr, stdin, stdout

`stdin`

FILE * correspondente ao stream de entrada padrão

`stdout`

FILE * correspondente ao stream de saída padrão

`stderr`

FILE * correspondente ao stream de erro padrão