DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

I/O
Operações sobre ficheiros

Preparação para exercício 4
Revisão de IAED + alguma matéria nova

DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Abordagem 1:
Trabalhar com ficheiros usando as funções da stdio

Operações sobre ficheiros

- Até este momento fizemos sempre leituras do stdin e escrevemos sempre para o stdout. Vamos ver agora como realizar estas operações sobre ficheiros.
- Como abrir um ficheiro?

```
FILE *fp;

fp=fopen("tests.txt","r");
```

Ponteiro para estrutura que representa o ficheiro aberto

Modo de abertura do ficheiro. Neste caso estamos a abrir o ficheiro em modo de leitura

Operações sobre ficheiros

- Até este momento fizemos sempre leituras do stdin e escrevemos sempre para o stdout. Vamos ver agora como realizar estas operações sobre ficheiros
- Como abrir um ficheiro?

Modos de abertura

r – abre para leitura (read)
w – abre um ficheiro vazio para escrita (o ficheiro não precisa de existir)
a – abre para acrescentar no fim ("append" ; ficheiro não precisa de existir)
r+ – abre para escrita e leitura; começa no início; o ficheiro tem de existir
w+ – abre para escrita e leitura (tal como o "w" ignora qualquer ficheiro que exista com o mesmo nome, criando um novo ficheiro)
a+ – abre para escrita e leitura (output é sempre colocado no fim)
…mas há mais

Exemplo

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

int main()
{
  FILE *fp;

  fp = fopen("teste.txt", "r");
  if (fp == NULL) {
    printf("teste.txt: No such file or directory\n");
    exit(1);
  }

  return 0;
}
```

Se não conseguir abrir, fp fica igual a NULL

Exemplo

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

int main()
{
  FILE *fp;

  fp = fopen("teste.txt", "r");
  if (fp == NULL) {
    perror("teste.txt");
    exit(1);
  }

  return 0;
}
```

Escreve a mesma mensagem de erro.

perror() escreve no "standard error" (stderr) a descrição do último erro encontrado na chamada a um sistema ou biblioteca.

Exemplo 2

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

int main()
{
  FILE *fp;

  fp = fopen("teste.txt", "r");
  if (fp == NULL) {
    perror("teste.txt");
    exit(1);
  }

  fclose(fp);

  return 0;
}
```

Fecha o ficheiro

Exemplo 3

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

int main()
{
  FILE *fp;

  fp = fopen("teste.txt", "w");
  if (fp == NULL) {
    perror("teste.txt");
    exit(1);
  }

  fprintf(fp, "Hi file!\n");

  fclose(fp);

  return 0;
}
```

Escreve para um ficheiro

Exemplo 3

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

int main()
{
  FILE *fp;

  fp = fopen("teste.txt", "w");
  if (fp == NULL) {
    perror("teste.txt");
    exit(1);
  }

  fputs("Hi file!", fp);

  fclose(fp);

  return 0;
}
```

Escreve para um ficheiro (alternativa)

Exemplo 4

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

int main()
{
  FILE *myfile; int i;
  float mydata[100];
  myfile = fopen("info.dat", "r");
  if (myfile== NULL) {
    perror("info.dat");
    exit(1);
  }
  for (i=0;i<100;i++)
     fscanf(myfile,"%f",&mydata[i]);

  fclose(myfile);
  return 0;
}
```

Lê um conjunto de 100 floats guardados num ficheiro

Exemplo 5

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

int main()
{
  FILE *myfile; int i;
  myfile = fopen("info.dat", "a");

  for (i=0;i<100;i++)
     fprintf(myfile,"%d\n",i);

  fclose(myfile);
  return 0;
}
```

Adiciona um conjunto de 100 inteiros ao fim de um ficheiro

Exemplo 6: Ler matriz guardada em ficheiro

```
DoubleMatrix2D *readMatrix2dFromFile(FILE *f, int l, int c) {
  int i, j; double v;
  DoubleMatrix2D *m;

  m = dm2dNew(l, c);

  for (i = 0; i < l; i++) {
    for (j = 0; j < c; j++) {
      if (fscanf(f, "%lf", &v)
        dm2dFree(m);
        return NULL;
      }
      dm2dSetEntry(m, i, j, v);
    }
  }
  return m;
}
```

Lê cada valor no ficheiro

Exercício: Escrever matriz para ficheiro

- Pegar no matrix2d.c
- Criar nova função que imprime matriz em ficheiro passado como argumento
- Chamar esta função sobre ficheiro “out.txt” aberto no final do *main* do heatSim

void dm2dPrintToFile(DoubleMatrix2D *matrix, FILE *f);

O cursor

- Para qualquer ficheiro aberto, é mantido um cursor
- O cursor avança automaticamente com cada byte lido ou escrito

- Para sabermos em que posição estamos, usar função ***ftell***

```
long ftell(FILE *stream);
```

- Para repor o cursor noutra posição, usar função ***fseek***
  - Por exemplo, colocar cursor no início ou final do ficheiro

```
int fseek(FILE *stream, long offset, int whence);
```

Escritas são imediatamente persistentes?

- Após escrita em ficheiro, essa escrita está garantidamente persistente no disco?
  - Nem sempre!
  - Para optimizar o desempenho, escritas são propagadas para disco tardiamente.
- Função ***fflush*** permite ao programa forçar que escritas feitas até agora sejam persistidas em disco
  - Função só retorna quando houver essa garantia
  - Função demorada, usar apenas quando necessário

```
int fflush(FILE *stream);
```


DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Abordagem 2: Trabalhar com ficheiros usando as funções da API do sistema de ficheiros do Unix

O que ganho/perco?

Prós:

- Em geral, são funções de mais baixo nível, logo permitem maior controlo
- Algumas operações sobre ficheiros só estão disponíveis através desta API

Contras:

- Normalmente, programa que usa *stdio* é mais simples e optimizado
  - Discutiremos mais à frente em SO porque é que stdio é mais optimizado

Sistema de Ficheiros do Unix

Operações	Genéricas	Linux
Simples	Fd := Abrir (Nome, Modo) Fd := Criar (Nome, Protecção)	int open(const char *path, int flags, mode_t mode)
	Fechar (Fd)	int close(int fd)
Ficheiros Abertos	Ler (Fd, Tampão, Bytes)	int read(int fd, void *buffer, size_t count)
	Escrever (Fd, Tampão, Bytes)	int write(int fd, void *buffer, size_t count)
	Posicionar (Fd, Posição)	int lseek(int fd, off_t offset, int origin)
Complexas	Criar link (Origem, Destino)	int symlink(const char *oldpath, const char *newpath) int link(const char *oldpath, const char *newpath)
	Mover (Origem, Destino)	int rename(const char *oldpath, const char *newpath)
	Apagar link (Nome)	int unlink(const char *path)
		int dup(int fd), int dup2(int oldfd,int newfd)
	LerAtributos (Nome, Tampão)	int stat(const char *path, struct stat *buffer)
	EscreverAtributos (Nome, Atributos)	int fcntl(int fd,int cmd,struct flock *buffer) int chown(const char *path, uid_t uid, gid_t gid) int chmod(const char *path, mode_t mode)
Ficheiros em memória	MapearFicheiro(Fd,pos,endereço,dim)	void *mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset)
	DesMapearFicheiro(endereço,dim)	int munmap(void *addr, size_t len)
Directórios	ListaDir (Nome, Tampão)	int readdir(int fd, struct dirent *buffer, unsigned int count)
	MudaDir (Nome)	int chdir(const char *path)
	CriaDir (Nome, Protecção)	int mkdir(const char *path, mode_t mode)
	RemoveDir(Nome)	int rmdir(const char *path)
Sistemas de Ficheiros	Montar (Directório, Dispositivo)	int mount(const char *device, const char *path, const char *fstype, unsigned long flags, const void *data)
	Desmontar (Directório)	int umount(const char *path)

Sistema de Ficheiros do Unix: Depois desta aula

- Estudar as *man pages* destas funções
- Em particular, as funções *unlink* e *rename* serão certamente úteis no 4º exercício do projeto

- Analisaremos o funcionamento interno destas funções dentro de algumas semanas nas teóricas de SO