Arquitetura de Computadores 2017-18

Ficha 8

Tópicos: *IO e ficheiros em C.*

Introdução

As operações envolvendo periféricos estão normalmente reservadas ao SO (que executa com o CPU em modo supervisor), como forma de permitir a sua correta operação e a partilha entre múltiplos processos. Em particular, todas as operações de IO são conseguidas com pedidos ao SO, mesmo pelas bibliotecas *standard* das linguagens. Por outro lado, a interface oferecida pelo SO e, ainda mais, as interfaces oferecidas pelas linguagens facilitam em muito a programação e o processamento das entradas e saídas.

1. Canais "bufferizados"

Verifique o código C seguinte que pretende afixar a passagem de tempo, segundo a segundo. Compile e execute o programa.

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

#define NSEC 10

int main( ) {
  printf("Contando tempo: ");
  for ( int i=0; i<NSEC; i++ ) {
    printf(" %d", i );
    sleep(1);
  }
  putchar('\n');
  return 0;
}
```

Aguarde os 10 seg. que são contados pelo programa. Confirme que todo o texto só aparece no ecrã no fim do tempo. Justifique o comportamento observado.

a) Sabendo que nos canais para "ficheiros" que sejam o ecrã, estes só passam realmente a informação escrita ao SO quando os buffers internos enchem ou a quando de um fim-de-linha ('\n'), garanta que o programa afixa a passagem de cada segundo.
b) Sabendo que existe a função fflush(), que obriga um canal (*stream*) a despejar todo o seu conteúdo para o SO, garanta que o programa afixa a passagem de cada segundo (desta vez sem mudanças de linha pelo meio).
c) Se não houver vantagem em estar constantemente a atualizar o ecrã (ou um ficheiro), deve usar a função fflush? (pense em termos de desempenho)

2. Ficheiros de texto e binários

Os ficheiros podem conter quaisquer bytes mas, mesmo os binários, têm muitas vezes texto que pode ser interessante ver no ecrã. Implemente um programa que, para qualquer ficheiro indicado na linha de comando, mesmo que binário, afixa no ecrã o texto que for encontrado. Considere como texto apenas sequencias de pelo menos 5 caracteres do conjunto de códigos visíveis do ASCII e ainda mudanças de linha ('\n') e tabulação ('\t'). Tal significa que não vamos considerar os caracteres acentuados, entre outros, como visíveis.

Sugestão: pode usar a função fgetc que permite ler byte-a-byte o ficheiro. Tenha em atenção que esta devolve o byte lido (que pode ser um char) convertido para um int, ou então uma constante EOF, que assinala que já não há mais bytes para ler. Para testar os caracteres lidos pode usar a função isprint que verifica se o argumento representa um caracter ASCII visível (texto). Use como ponto de partida o código seguinte:

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

#define SZ 5  // tem de ser pelo menos estes caracteres seguidos para afixar


int main( int argc, char *argv[] ) {
    FILE *f;

    // Verificar os argumentos da linha de comando
    if (argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    // Tentar abrir o ficheiro para leitura
    f=fopen (argv[1], "r");
    if (f== NULL) {
        printf("Invalid file: %s\n", argv[1]);
        return 1;
    }
// TODO: completar

    return 0;
}
```

Mais informação

Veja a documentação sobre as funções `fgetc`, `putchar`, `fflush` e outras no seu livro de C ou nos manuais do Linux (exemplo: `man fgetc`).