Programação Sistema com Ficheiros

Chamadas ao Sistema e Bibliotecas de Funções

- Os sistemas operativos não permitem que os programas acedam directamente aos vários recursos do sistema: ficheiros, teclado, ecrã, entre outros. Desta forma garante-se a integridade destes recursos, apenas se fornece o acesso ao recurso a determinados processos, optimiza-se a gestão dos vários recursos.
- Para manipularem estes elementos os programa têm de efectuar pedidos ao sistema operativo para tal. Um programa podem pedir por exemplo para o sistema operativo acrescentar uma linha x a um ficheiro y.
- As chamadas ao sistema podem ser por vezes um pouco complicadas uma vez que são pedidos directos ao sistema operativo.
- Por este motivo existem bibliotecas intermediárias que se encarregam elas próprias de efectuar os pedidos ao sistema operativo. Estas bibliotecas são constituidas por um conjunto de funções que podem ser utilizadas pelos vários programas.
- Para além de serem mais simples de utilizar que as chamadas ao sistema, as bibliotecas também podem tornar os programa portáveis entre plataformas (Windows e Linux) caso a biblioteca exista nas várias plataformas.

Utilização da Standard Input Output Library (stdio) para a manipulação de ficheiros

- O desenvolvimento de programas com recurso a bibliotecas existentes em várias plataformas, como é o caso da Standard Input Output Library (stdio), permite que os programas corram nessas diferentes plataforma. Os programas seguintes correm quer no Windows quer no Linux.
- Programa que abre um ficheiro, lê o primeiro caracter, escreve o caracter lido no ecrã e finalmente fecha o ficheiro.

```
#include <stdio.h>

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

  f = fopen(argv[1], "r");
  c = fgetc(f);
  fputc(c, stdout);
  fclose( f );
```

```
  return 0;
}
```

- Programa que mostra o conteúdo de um ficheiro utilizando as funções fgetc e fputc. A leitura do ficheiro é efectuada caracter a caracter.

```
#include <stdio.h>

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

  f = fopen(argv[1], "r");
  c = fgetc(f);
  while( c != EOF ){
    fputc(c, stdout);
    c =  fgetc(f);
  }
  fclose( f );

  return 0;
}
```

- Igual ao anterior mas com o código mais compactado.

```
#include <stdio.h>

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

  f = fopen(argv[1], "r");
  while( (c = fgetc(f) ) != EOF ){
    fputc(c, stdout);
  }
  fclose( f );

  return 0;
}
```

- Programa que mostra o conteúdo de um ficheiro utilizando as funções fgets e fputs. A leitura do ficheiro é efectuada linha a linha.

```
#include <stdio.h>
#define MAXLIN 80

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

  f = fopen(argv[1], "r");
```

```
  while( fgets(l, MAXLIN, f) != NULL ){
    fputs(l, stdout);
  }
  fclose(f);

  return 0;
}
```

- Programa que mostra o conteúdo de um ficheiro utilizando as funções fread e fwrite. A leitura do ficheiro é efectuada bloco a bloco.

```
#include <stdio.h>
#define BUFSIZE 5

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

  f = fopen(argv[1], "r");
  while( (n = fread(buffer, sizeof(char), BUFSIZE, f) ) > 0 ){
    fwrite(buffer, sizeof(char), n, stdout);
  }
  fclose(f);

  return 0;
}
```

- As funções fprintf e fscanf. Estas funções são semelhantes às funções printf e scanf, mas são mais gerais, pois permitem a escrita ou leitura em qualquer ficheiro. Este programa lê o ficheiro palavra a palavra e escreve cada uma destas numa linha diferente do ecrã.

```
#include <stdio.h>
#define MAXWORD 20

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

  f = fopen(argv[1], "r");
  while( fscanf( f, "%s", w ) != EOF ){
    fprintf( stdout, "%s\n", w );
  }
  fclose(f);

  return 0;
}
```

Utilização de chamadas ao sistema para a manipulação de ficheiros no UNIX

- Abertura e leitura de um ficheiro. Descritor de um ficheiro. Este programa escreve no ecrã o conteúdo do ficheiro fornecido como parametro (semelhante ao comando cat).

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#define BUFSIZE 512

int main(int argc, char *argv[]){
  int fd;
  char buffer[BUFSIZE];
  int numBytes;

  fd = open(argv[1], O_RDONLY);
  while ( (numBytes = read( fd, buffer, BUFSIZE) ) > 0 )
    write (STDOUT_FILENO, buffer, numBytes);
  close(fd);

  return 0;
}
```

Última revisão efectuada em 05/12/18