Sistemas de Operação (2018/2019)
Ficha 3

Q1. Considere o seguinte programa que recebe duas strings na linha de comando (`argv[1]` e `argv[2]`) e realiza operações com elas com a API de "strings" da Biblioteca Standard do C (clib). Compile-o e experimente-o.

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

#define MAX_STR_SIZE  64

int main(int argc, char* argv[]) {
  char* p1 = (char*)malloc(MAX_STR_SIZE * sizeof(char));
  char* p2 = (char*)malloc(MAX_STR_SIZE * sizeof(char));

  int result = strcmp(argv[1], argv[2]);

  if (result == 0)
    printf("the strings are the same\n");
  else if (result < 0)
    printf("%s < %s\n", argv[1], argv[2]);
  else
    printf("%s > %s\n", argv[2], argv[1]);

  strcpy(p1, argv[1]);
  strcpy(p2, argv[2]);
  printf("p1 holds:%s\n", p1);
  printf("p2 holds:%s\n", p2);

  strcat(p1,p2);
  printf("p1 holds:%s\n", p1);
  strcat(p2,p1);
  printf("p2 holds:%s\n", p2);

  return EXIT_SUCCESS;
}
```

Faça `man 3 string` para ver a API completa. Com base neste exemplo, escreva agora um programa que:

- recebe uma string na linha de comando e a transforma numa string equivalente mas com todos os caracteres em minúsculas;

- recebe duas strings na linha de comando e indica se a primeira ocorre na segunda;
- recebe duas strings na linha de comando e indica quantas vezes a primeira ocorre na segunda.

Q2. Considere o programa `naughty.c` que usa a API de strings. Experimente-o usando strings com diversos tamanhos na linha de comando.

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

int f(char* content) {
  char str[8];
  int result = 2;
  (void)strcpy(str, content);
  return result;
}

int main(int argc, char* argv[]) {
  if (argc == 2)
    printf("result = %d\n", f(argv[1]));
  return EXIT_SUCCESS;
}
```

Compile-o e execute-o com diferentes strings, e.g.:

```
$ ./naughty a
$ ./naughty ab
$ ./naughty abc
$ ./naughty abcd
$ ./naughty abcde
$ ./naughty abcdef
$ ./naughty abcdefg
$ ./naughty abcdefgh
$ ./naughty abcdefghi
$ ./naughty abcdefghij
```

Como explica a diferença de resultados obtidos? Faça um desenho da pilha de execução do programa para melhor visualizar o problema. Como poderia corrigir o programa?

Q3. Usando a API de manipulação de ficheiros da Biblioteca Standard do C (`fopen`, `fclose`, `fseek`, `fread` e `fwrite`), escreva um programa `mycat` que:

- recebe como argumento o nome de um ficheiro e imprime o seu conteúdo (semelhante ao comando `cat` com 1 argumento);
- recebe como argumento os nomes de vários ficheiros e imprime o conteúdo de todos os eles sequencialmente (semelhante ao comando `cat` com vários argumentos).

```
$ cat > file1
This is a test
^D
$ cat > file2
Another test
^D
$ cat > file3
And yet another
^D
$ ./mycat file1
This is a test
$ ./mycat file1 file3
This is a test
And yet another
$ ./mycat file1 file2 file3
This is a test
Another test
And yet another
```

Q4. Crie um programa `chcase` que recebe como argumentos um nome de um ficheiro e uma "flag", e envie para o standard output o conteúdo do ficheiro dado:

- com todas as letras maiúsculas se a flag for `-u`;
- com todas as letras minusculas se a flag for `-l`;
- inalterado, se nenhuma das anteriores.

Por exemplo:

```
$ cat > teste.txt
Ads fTsfsdsR DSda BVHGIsdssdeSds
Dfcdfd 45343f rerTEuk
qqfFGfhuymIOu 95r342
^D
$ ./chcase -u teste.txt
ADS FTSFSDSR DSDA BVHGISDSSDESDS
DFCDFD 45343F RERTEUK
```

```
QQFFGFHUYMIOU 95R342
$ ./chcase -l teste.txt
ads ftsfsdsr dsda bvhgisdssdesds
dfcdfd 45343f rerteuk
qqffgfhuymiou 95r342
$ ./chcase teste.txt
Ads fTsfsdsR DSda BVHGIsdssdeSds
Dfcdfd 45343f rerTEuk
qqfFGfhuymIOu 95r342
```

Sugestão: faça `man tolower` e `man toupper` para ver funções da clib que podem ser relevantes.

Q5. Escreva um programa `mygrep` que dada uma string e um ficheiro na linha de comando imprima todas as ocorrências da string no ficheiro, indicando a linha e a coluna do texto onde começam, e.g.:

```
$ ./mygrep agulha palheiro.txt
[2:17],[5:2],[23:7]
```

Q5. Escreva uma programa que dados dois ficheiros `file1` e `file2` como argumentos, copie o conteúdo de `file1` para `file2`. Se o segundo não existir deverá ser criado. Se existir o seu conteúdo será reescrito. Esta é a forma como funciona o comando `cp` da Bash shell.

```
$ cat > file1
This is a test
^D
$ ./mycp file1 file2
$ cat file2
This is a test
$ cat > file3
Another test
^D
$ ./mycp file3 file2
$ cat file2
Another test
```

Q6. Escreva um programa `mywc` que dado um ficheiro de texto como argumento escreva:

- o número de caracteres, se a flag `-c` for usada;
- o número de palavras, se a flag `-w` for usada;

- o número de linhas, se a flag `-l` for usada;
- o número de caracteres por defeito.

```
$ cat > file.txt
This is a test
^D
$ ./mywc -c file.txt
15
$ ./mywc -w file.txt
4
$ ./mywc -l file.txt
1
$ ./mywc file.txt
15
```