Guião aula05 - Gil Guedes 125031

1 - a - Para um único arquivo, lê e imprime cada linha com "->" antes.

b / c -

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

/* SUGESTÂO: utilize as páginas do manual para conhecer mais sobre as
funções usadas:
man fopen
man fgets
*/

#define LINEMAXSIZE 80 /* or other suitable maximum line size */

int main(int argc, char *argv[])
{
   FILE *fp = NULL;
   char line [LINEMAXSIZE];

   /* Validate number of arguments */
   if ( argc < 2 )
   {
       printf("USAGE: %s fileName [fileName2 ...]\n", argv[0]);
       return EXIT_FAILURE;
   }
   for (int i = 1; i < argc; i++)
   {
       /* Open file */
       errno = 0;
       fp = fopen(argv[i], "r");
       if ( fp == NULL )
       {
           perror ("Error opening file");
           printf("File: %s\n", argv[i]);
           continue;
       }

   size_t line_number = 1;
       /* Read all the lines of the file */
   while (fgets(line, sizeof(line), fp) != NULL) {
       /* fgets lê o '\n' que é o final da linha */
     printf("%lu -> %s", line_number++, line);
   }
   fclose(fp);
   }
   return EXIT_SUCCESS;
}
```

d -

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

/* SUGESTÂO: utilize as páginas do manual para conhecer mais sobre as
funções usadas:
man fopen
man fgets
*/

#define LINEMAXSIZE 1024 /* or other suitable maximum line size */

int main(int argc, char *argv[])
{
   FILE *fp = NULL;
   char line [LINEMAXSIZE];

   /* Validate number of arguments */
   if ( argc < 2 )
   {
       printf("USAGE: %s fileName [fileName2 ...]\n", argv[0]);
       return EXIT_FAILURE;
   }

   for (int i = 1; i < argc; i++)
   {
       /* Open file */
       errno = 0;
       fp = fopen(argv[i], "r");
       if ( fp == NULL )
       {
           perror ("Error opening file");
           printf("File: %s\n", argv[i]);
           continue;
       }
   size_t line_number = 1;
       /* Read all the lines of the file */
   while (fgets(line, sizeof(line), fp) != NULL) {
       /* fgets lê o '\n' que é o final da linha */
     printf("%lu -> %s", line_number, line);

     int len = strlen(line);
     if (len > 0 && (line[len-1] == '\n' || len < sizeof(line)-1)) {
         line_number++;
     }
   }
   fclose(fp);
   }
   return EXIT_SUCCESS;
}
```

2 - a - O código ordena os números.

b - i -

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

/* SUGESTÂO: utilize as páginas do manual para conhecer mais sobre as
funções usadas:
man fgets
man qsort
*/

#define MAX_NUMBERS 100
#define LINE_SIZE 256

int compareInts(const void *px1, const void *px2)
{
   int x1 = *((int *)px1);
   int x2 = *((int *)px2);
   return(x1 < x2 ? -1 : x1 == x2 ? 0 : 1);
}

int main(int argc, char *argv[])
{
   FILE *fp = NULL;
   char line[LINE_SIZE];
   int numbers[MAX_NUMBERS];
   int count = 0;
   int i;

   /* Validate number of arguments */
   if (argc != 2)
   {
       printf("USAGE: %s fileName\n", argv[0]);
       return EXIT_FAILURE;
   }

   /* Open the file */
   errno = 0;
   fp = fopen(argv[1], "r");
   if (fp == NULL)
   {
       perror("Error opening file");
       return EXIT_FAILURE;
   }

   /* Read numbers from file using fgets() */
   while (fgets(line, sizeof(line), fp) != NULL && count < MAX_NUMBERS)
   {
       line[strcspn(line, "\n")] = 0;
```

```
        numbers[count] = atoi(line);
        count++;
    }

    fclose(fp);

    /* Check if we read any numbers */
    if (count == 0)
    {
        printf("No numbers found in the file.\n");
        return EXIT_SUCCESS;
    }

    /* Sort the numbers using qsort */
    qsort(numbers, count, sizeof(int), compareInts);

    /* Print the sorted numbers */
    printf("Sorted numbers from file %s:\n", argv[1]);
    for (i = 0; i < count; i++)
    {
        printf("%d\n", numbers[i]);
    }

    return EXIT_SUCCESS;
}
```

b - ii -

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

/* SUGESTÂO: utilize as páginas do manual para conhecer mais sobre as
funções usadas:
man fscanf
man qsort
*/

#define MAX_NUMBERS 100

int compareInts(const void *px1, const void *px2)
{
    int x1 = *((int *)px1);
    int x2 = *((int *)px2);
    return(x1 < x2 ? -1 : x1 == x2 ? 0 : 1);
}

int main(int argc, char *argv[])
{
    FILE *fp = NULL;
    int numbers[MAX_NUMBERS];
    int count = 0;
    int number;
```

```c
    int i;

    /* Validate number of arguments */
    if (argc != 2)
    {
        printf("USAGE: %s fileName\n", argv[0]);
        return EXIT_FAILURE;
    }

    /* Open the file */
    errno = 0;
    fp = fopen(argv[1], "r");
    if (fp == NULL)
    {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    /* Read numbers from file using fscanf() */
    while (fscanf(fp, "%d", &number) == 1 && count < MAX_NUMBERS)
    {
        numbers[count] = number;
        count++;
    }

    fclose(fp);

    /* Check if we read any numbers */
    if (count == 0)
    {
        printf("No numbers found in the file.\n");
        return EXIT_SUCCESS;
    }

    /* Sort the numbers using qsort */
    qsort(numbers, count, sizeof(int), compareInts);

    /* Print the sorted numbers */
    printf("Sorted numbers from file %s:\n", argv[1]);
    for (i = 0; i < count; i++)
    {
        printf("%d\n", numbers[i]);
    }

    return EXIT_SUCCESS;
}
```

C -

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

/* SUGESTÂO: utilize as páginas do manual para conhecer mais sobre as
funções usadas:
man system
man date
*/

int main(int argc, char *argv[]) {
   char text[1024];
   FILE *logFile;
   time_t rawTime;
   struct tm *timeInfo;
   char timeString[80];

   do
   {
       printf("Command: ");
       scanf("%1023[^\n]%*c", text);

       /* Get current time */
       time(&rawTime);
       timeInfo = localtime(&rawTime);
             strftime(timeString,  sizeof(timeString),  "%Y-%m-%d  %H:%M:%S",
timeInfo);

       /* Escrever no log */
       logFile = fopen("command.log", "a");
       if (logFile != NULL) {
           fprintf(logFile, "[%s] %s\n", timeString, text);
           fclose(logFile);
       } else {
           printf("Warning: Could not open log file for writing\n");
       }

          /* system(const char *command) executes a command specified in
command
           by calling /bin/sh -c command, and returns after the command has
been
           completed.
       */
       if(strcmp(text, "end")) {
          printf("\n * Command to be executed: %s\n", text);
          printf("---------------------------------\n");
          system(text);
          printf("---------------------------------\n");
       }
   } while(strcmp(text, "end"));
```

```
  printf("-----------The End---------------\n");

  return EXIT_SUCCESS;
}
```

4 - a - Mostra o conteúdo do diretório dado no argumento. ou seja, é um ‘ls’ command.

b -

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <string.h>

/* SUGESTÂO: utilize as páginas do manual para conhecer mais sobre as
funções
 usadas: man opendir man readdir
*/

void listDir(char dirname[]) {
 DIR *dp;
 struct dirent *dent;

 dp = opendir(dirname);
 if (dp == NULL) {
   perror("Error opening directory");
   return;
 }

 dent = readdir(dp);
 while (dent != NULL) {
   if (dent->d_name[0] != '.') /* do not list hidden dirs/files */
   {
     struct stat path_stat;
     char fullPath[1024];
     snprintf(fullPath, sizeof(fullPath), "%s/%s", dirname, dent->d_name);

     if (stat(fullPath, &path_stat) != 0) {
       perror("Error executing stat");
       return;
     }

     char type = ' ';
     if (S_ISDIR(path_stat.st_mode))
       type = 'd';

     printf("%c %s/%s\n", type, dirname, dent->d_name);
   }

   dent = readdir(dp);
```

```
 }
  closedir(dp);
}

int main(int argc, char *argv[]) {
 if (argc != 2) {
   fprintf(stderr, "Usage: %s base_directory\n", argv[0]);
   return EXIT_FAILURE;
 }

 listDir(argv[1]);

 return EXIT_SUCCESS;
}
```

C -