SCC0222 - Laboratório de Introdução à Ciência de Computação I

Arquivos

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Arquivos [1]

- Podemos manipular arquivos em C usando um ponteiro para a struct FILE
- Para usar um arquivo é preciso abri-lo com a função fopen(), manipular sua entrada/saída como desejado e fechá-lo com fclose() ao terminar
- Tudo isso é feito com funções da biblioteca *stdio.h*
  - E outras bibliotecas que forem necessárias

Arquivos [1, 2]

- ➔ *fopen(filename, mode)* é responsável por abrir um arquivo "*filename*" e retornar um ponteiro para *FILE*
- ➔ O parâmetro *mode* é quem define quais operações são permitidas sobre aquela corrente de dados
- ➔ A corrente de dados retornada é considerada totalmente *bufferizada* desde que não pertença a um dispositivo interativo

Arquivos [2]

- Modos
  - “r” - read
    - Abre o arquivo para operações de entrada (leitura)
    - O arquivo precisa existir

Arquivos [2]

- Modos
  - “w” - write
    - Cria um arquivo em branco para operações de saída (escrita)
    - Se já existe um arquivo com o nome passado, seu conteúdo é deletado e o arquivo é tratado como um novo

Arquivos [2]

- Modos
  - “a” - append
    - Abre um arquivo para operações de saída (escrita)
    - Operações de saída sempre escrevem dados ao final do arquivo, expandindo ele
    - Operações de reposição são ignoradas (ex: fseek)
    - Se o arquivo não existir, ele é criado

Arquivos [2]

- Modos
  - “r+” - read/update
    - Abre o arquivo para operações de entrada e saída
    - O arquivo precisa existir

Arquivos [2]

- Modos
  - “w+” - write/update
    - Cria um arquivo em branco para operações de entrada e saída
    - Se já existe um arquivo com o nome passado, seu conteúdo é deletado e o arquivo é tratado como um novo

Arquivos [2]

- Modos
  - “a+” - append/update
    - Abre arquivo para operações de entrada e saída
    - Operações de saída sempre escrevem dados ao final do arquivo, expandindo ele
    - Operações de reposição (ex: fseek)afetam apenas operações de leitura. Escrita volta ao fim
    - Se o arquivo não existir, ele é criado

Arquivos [2]

- Modos
  - “b” - binary
    - Todos os exemplos anteriores tratam de arquivos de texto.
    - Ao adicionar a letra “b” ao final de cada uma das anteriores (ou antes do símbolo de “+”) o arquivo é tratado como binário

Arquivos [2]

- Modos
  - “x” - APENAS NO C2011
    - Ao ser adicionado a uma operação com “w”, força a função a falhar caso o arquivo já exista
      - Impede a sobrescrita!

Arquivos [2]

```c
int main ()
{
  FILE * pFile;
  pFile = fopen ("myfile.txt","w");
  if (pFile!=NULL)
  {
    fputs ("fopen example",pFile);
    fclose (pFile);
  }
  return 0;
}
```

Arquivos [5]

- fputs(*str*, *stream*)
  - Escreve a string apontada por *str* no arquivo *stream*
  - Copia até encontrar o caractere null ‘\0’
    - Não copia o ‘\0’
  - Difere do *puts()* ao deixar definir a *stream* e ao não escrever automaticamente ‘\n’ ao fim

Arquivos [1]

```c
int main() {
   FILE *pFile;
   char buffer [100];
   pFile = fopen ("aizen.txt" , "r");
   if (pFile == NULL) perror ("Error opening file");
   else {
     while ( ! feof (pFile) ) {
       if ( fgets (buffer , 100 , pFile) == NULL ) break;
       fputs (buffer , stdout);
     }
     fclose (pFile);
   }
   return 0;
}
```

Arquivos [3]

- fgets(*str*, *num*, *stream*)
  - Lê caracteres de uma *stream* e salva eles como uma string em C em *str* até (*num*-1) caracteres serem lidos
    - OU encontrar um ‘\n’ OU EOF
    - O ‘\n’ é incluído na string
  - ‘\0’ é adicionado ao fim da string automaticamente

Arquivos [4]

```c
int main (){
  FILE * pFile;
  pFile = fopen ( "example.txt" , "wb" );
  fputs ( "This is an apple." , pFile );
  fseek ( pFile , 9 , SEEK_SET );
  fputs ( " sam" , pFile );
  fclose ( pFile );
  return 0;
}
```

Arquivos [4]

- fseek(*stream*, *offset*, *origin*)
  - Coloca o indicador de posição associado ao *stream* numa nova posição
  - Para *streams* abertas no modo binário, a nova posição é definida adicionando um *offset* para uma posição de referência especificada por *origin*

Arquivos [4]

- fseek(*stream*, *offset*, *origin*)
  - Para *streams* em modo texto, o *offset* deve ser ou 0 ou um valor retornado por uma chamada de *ftell()*
    - A origem deve ser necessariamente *SEEK_SET*
  - Outros valores usados dependem do suporte do sistema e da biblioteca usadas
  - Em *streams* abertas para *update*, a chamada de *fseek* permite trocar entre ler e escrever

Arquivos [4]

- fseek(*stream*, *offset*, *origin*)
  - Constantes de *origin*
    - SEEK_SET - Começo do arquivo
    - SEEK_CUR - Posição atual do ponteiro do arquivo
    - SEEK_END - Fim do arquivo

Arquivos [6]

```c
int main (){
  FILE * pFile;
  long size;
  pFile = fopen ("example.txt","rb");
  if (pFile==NULL) perror ("Error opening file");
  else  {
    fseek (pFile, 0, SEEK_END);   // non-portable
    size=ftell (pFile);
    fclose (pFile);
    printf ("Size of myfile.txt: %ld bytes.\n",size);
  }
  return 0;
}
```

Arquivos [6]

- ftell(*stream*)
  - Retorna o valor do indicador de posição da *stream*
  - Para binários é o número de bytes do começo do arquivo
  - Para textos, valor pode não ser significativo mas ainda pode ser usado para voltar à mesma posição depois usando *fseek()*

Arquivos [7]

```c
int main (){
   FILE * pFile;
   int n;
   char name [100];
   pFile = fopen ("myfile.txt","w");
   for (n=0 ; n<3 ; n++)   {
     puts ("please, enter a name: ");
     gets (name);
     fprintf (pFile, "Name %d [%-10.10s]\n",n+1,name);
   }
   fclose (pFile);
   return 0;
}
```

Arquivos [7]

```c
int main (){
    FILE * pFile;
    int n;
    char name [100];
    pFile = fopen ("myfileprintf.txt","r");
    while(fscanf(pFile, "Name %d %[^\n]",&n,name) != EOF){
        fgetc(pFile);
        printf("Name %d %s\n", n, name);
    }
    fclose (pFile);

    return 0;
}
```

Arquivos [7]

- fprintf(*stream*, format, ...)
  - Escreve a string apontada por *format* no *stream*
    - Segue os princípios do *printf*
- fscanf(*stream*, format, ...)
  - Lê de *stream* os dados e salva de acordo com a formatação de *format*
    - Segue os princípios do *scanf*

Arquivos [11, 12]

```c
#define VECTORSIZE 10
int main(){
    float vector[VECTORSIZE];
    FILE *fp;
    srand(time(NULL));
    for(int i = 0; i < VECTORSIZE; ++i)
        vector[i] = rand()/(float)(RAND_MAX/100);
    fp = fopen("data/numbers.bin","wb");
    fwrite(vector, sizeof(float), VECTORSIZE, fp);
    fclose(fp);
    return 0;
}
```

Arquivos [11, 12]

```
int main(){
    float *vector;
    int vectorSize;
    long fileSize;
    FILE *fp;
    fp = fopen("data/numbers.bin","rb");
    fseek(fp, 0, SEEK_END);
    fileSize = ftell(fp);
    fseek(fp, 0, SEEK_SET);
    vectorSize = fileSize/sizeof(float);
    vector = (float*) malloc(vectorSize*sizeof(float));
    fread(vector, sizeof(float), vectorSize, fp);
    for(int i = 0; i < vectorSize; ++i)
        printf("%f ", vector[i]);
    printf("\n");
    fclose(fp);
    return 0;
}
```

Arquivos [11, 12]

- fwrite(*ptr, size, count, stream*)
  - Escreve o dado apontado por *ptr* no *stream*
    - O quanto do dado é escrito é calculado pelo tamanho de cada elemento (*size*) vezes o *count*
- fread(*ptr, size, count, stream*)
  - Lê o dado do *stream* e salva em *ptr*
    - O quanto do dado é lido é calculado pelo tamanho de cada elemento (*size*) vezes o *count*

Referências

1. http://www.cplusplus.com/reference/cstdio/FILE/
2. http://www.cplusplus.com/reference/cstdio/fopen/
3. http://www.cplusplus.com/reference/cstdio/fgets/
4. http://www.cplusplus.com/reference/cstdio/fseek/
5. http://www.cplusplus.com/reference/cstdio/fputs/
6. http://www.cplusplus.com/reference/cstdio/ftell/
7. http://www.cplusplus.com/reference/cstdio/fprintf/
8. http://www.cplusplus.com/reference/cstdio/fscanf/
9. https://www.geeksforgeeks.org/basics-file-handling-c/
10. https://www.geeksforgeeks.org/c-program-merge-contents-two-files-third-file/
11. http://www.cplusplus.com/reference/cstdio/fwrite/
12. http://www.cplusplus.com/reference/cstdio/fread/