Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

Arquivos em C – Parte 2

Profa Rosana Braga

Material preparado pela profa
Silvana Maria Affonso de Lara

1º semestre de 2010

Arquivos em Disco

- Abrindo e Fechando um Arquivo
  - fopen, exit, fclose
- Lendo e Escrevendo Caracteres em Arquivos
  - putc, getc, feof
- Outros Comandos de Acesso a Arquivos
  - *Arquivos pré definidos*
  - ferror e perror, fgets, fputs, fread
  - fwrite, fseek, rewind, remove
- Fluxos Padrão
  - fprintf, fscanf

Escrevendo / Lendo Caracteres em Arquivos

putc

- Escreve um caractere no arquivo
- Protótipo: **int** putc (int ch, FILE *fp);

getc

- Retorna um caractere lido do arquivo
- Protótipo: **int** getc (FILE *fp);

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp;
char string[100];
int i;
if((fp = fopen("arquivo.txt","w")) == NULL) {
/* Arquivo ASCII, para escrita */
        printf( "Erro na abertura do arquivo");
        exit(0);
}
printf("Entre com a string a ser gravada no
    arquivo:");
gets(string);
for(i=0; string[i]; i++) putc(string[i], fp);
    /* Grava a string, caractere a caractere */
fclose(fp);
}
```

Escrevendo / Lendo Caracteres em Arquivos

feof

- EOF ("End of file") indica o fim de um arquivo. Às vezes, é necessário verificar se um arquivo chegou ao fim. A função feof retorna não-zero se o arquivo chegou ao EOF, caso contrário retorna zero. Seu protótipo é:

```
int feof (FILE *fp);
```

- Outra forma de se verificar se o final do arquivo foi atingido é comparar o caractere lido por getc com EOF.

Exemplo: Leitura de Arquivo

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *fp;
char c;
if((fp = fopen("arquivo.txt","r")) == NULL){
    /* Arquivo ASCII, para leitura */
      printf( "Erro na abertura do arquivo");
      exit(0);
}
while((c = getc(fp) ) != EOF)
        /* Enquanto não chegar ao final do
    arquivo*/
      printf("%c", c);    /* imprime o caracter lido
    */
fclose(fp);
}
```

Exemplo

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main()
{
FILE *p;
char c, str[30], frase[80] = "Este e um arquivo chamado: ";
int i;
/* Le um nome para o arquivo a ser aberto: */
printf("\n\n Entre com um nome para o arquivo:\n");
gets(str);
if (!(p = fopen(str,"w"))){ /* programa aborta automaticamente */
        printf("Erro! Impossivel abrir o arquivo!\n");
       exit(1);
}
```

Exemplo

```
/* Se nao houve erro, imprime no arquivo e o fecha ...*/
strcat(frase, str);
for (i=0; frase[i]; i++)
      putc(frase[i], p);
fclose(p);
/* Abre novamente para leitura */
p = fopen(str,"r");
while (!feof(p)) {
       /* Enquanto não se chegar no final do arquivo */
       c = getc(p);      /* Le um caracter no arquivo */
      printf("%c",c);      /* e o imprime na tela */
}
fclose(p); /* Fecha o arquivo */
}
```

Arquivos pré-definidos

- Quando se começa a execução de um programa, o sistema automaticamente abre alguns arquivos pré-definidos:
  - stdin: dispositivo de entrada padrão (geralmente o teclado)
  - stdout: dispositivo de saída padrão (geralmente o vídeo)
  - stderr: dispositivo de saída de erro padrão (geralmente o vídeo)
  - stdaux: dispositivo de saída auxiliar (em muitos sistemas, associado à porta serial)
  - stdprn: dispositivo de impressão padrão (em muitos sistemas, associado à porta paralela)

Outros Comandos de Acesso a Arquivos

- Cada uma destas constantes pode ser utilizada como um ponteiro para FILE, para acessar os periféricos associados a eles. Desta maneira, pode-se, por exemplo, usar:

```
ch = getc(stdin);
```

- para efetuar a leitura de um caracter a partir do teclado, ou:

```
putc(ch, stdout);
```

para imprimi-lo na tela.

Outros Comandos de Acesso a Arquivos

fgets

- ler uma string num arquivo fgets() - protótipo é:

```
char *fgets (char *str, int tamanho, FILE *fp);
```

- A função recebe 3 argumentos:
  - a string a ser lida,
  - o limite máximo de caracteres a serem lidos e
  - o ponteiro para FILE, que está associado ao arquivo de onde a string será lida.

Outros Comandos de Acesso a Arquivos

fgets

- A função lê a string até que um caracter de nova linha seja lido ou tamanho-1 caracteres tenham sido lidos.
- Se o caracter de nova linha ('\n') for lido, ele fará parte da string, o que não acontecia com gets.
- A string resultante sempre terminará com '\0' (por isto somente tamanho-1 caracteres, no máximo, serão lidos).

Outros Comandos de Acesso a Arquivos

fputs

- Escreve uma string num arquivo.
- Protótipo: char *fputs (char *str, FILE *fp);

ferror

- Protótipo: int ferror (FILE *fp);
- A função retorna:

  - zero, se nenhum erro ocorreu;

  - um número diferente de zero se algum erro ocorreu durante o acesso ao arquivo.

Outros Comandos de Acesso a Arquivos

ferror

- ferror() se torna muito útil quando queremos verificar se cada acesso a um arquivo teve sucesso, de modo que consigamos garantir a integridade dos nossos dados. Na maioria dos casos, se um arquivo pode ser aberto, ele pode ser lido ou gravado. Porém, existem situações em que isto não ocorre. Por exemplo, pode acabar o espaço em disco enquanto gravamos, ou o disco pode estar com problemas e não conseguimos ler, etc.

Outros Comandos de Acesso a Arquivos

perror

- A função perror() (print error), cujo argumento é uma string que normalmente indica em que parte do programa o problema ocorreu.

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
FILE *pf;
char string[100];
if((pf = fopen("arquivo.txt","w")) ==NULL)
{
printf("\n Nao consigo abrir o arquivo ! ");
exit(1);
}
```

```
do
{
   printf("\nDigite uma nova string.
   Para terminar, digite <enter>: ");
   gets(string);
   fputs(string, pf);
   putc('\n', pf);
   if(ferror(pf))
   {
          perror("Erro na gravacao");
          fclose(pf);
          exit(1);
    }
} while (strlen(string) > 1);
fclose(pf);
}
```

Outros Comandos de Acesso a Arquivos

fread

- ler blocos de dados do arquivo. O protótipo de fread() é:

  unsigned fread (void *buffer, int numero_de_bytes,
  int count, FILE *fp);

- O buffer é a região de memória na qual serão armazenados os dados lidos. O *número_de_bytes* é o tamanho da unidade a ser lida. A variável *count* indica quantas unidades devem ser lidas

- Isto significa que o número total de bytes lidos é:

  numero_de_bytes X count.

Outros Comandos de Acesso a Arquivos

fread

- A função retorna o número de unidades efetivamente lidas. Este número pode ser menor que *count* quando o fim do arquivo for encontrado ou ocorrer algum erro.

- Quando o arquivo for aberto para dados binários, *fread* pode ler qualquer tipo de dados.

Outros Comandos de Acesso a Arquivos

fwrite

- Escreve blocos de dados num arquivo. Seu protótipo é:

  unsigned fwrite(void *buffer, int numero_de_bytes, int count, FILE *fp);

- A função retorna o número de itens escritos. Este valor será igual a *count* a menos que ocorra algum erro.

Outros Comandos de Acesso a Arquivos

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *pf;
    float pi = 3.1415;
    float pilido;

    if((pf = fopen("arquivo.bin", "wb")) == NULL)
    /* Abre arquivo binário para escrita */
    {
       printf("Erro na abertura do arquivo");
       exit(1);
    }
```

Outros Comandos de Acesso a Arquivos

```
if(fwrite(&pi, sizeof(float), 1,pf) != 1) /* Escreve a variável pi */
    printf("Erro na escrita do arquivo");
fclose(pf); /* Fecha o arquivo */
if((pf = fopen("arquivo.bin", "rb")) == NULL)
    /* Abre o arquivo novamente para leitura */
{
    printf("Erro na abertura do arquivo");
    exit(1);
}
if(fread(&pilido, sizeof(float), 1, pf) != 1)
    /* Le em pilido o valor da variável armazenada anteriormente */
    printf("Erro na leitura do arquivo");
printf("\n O valor de PI, lido do arquivo e': %f", pilido);
fclose(pf);
}
```

Outros Comandos de Acesso a Arquivos

- Uma das mais úteis aplicações de fread() e fwrite() envolve ler e escrever tipos de dados definidos pelo usuário, especialmente estruturas
- Exemplo:

```
struct struct_type{
        float balance;
        char name[80];
} cust;

fwrite(&cust, sizeof(struct struct_type),1,fp);
```

Outros Comandos de Acesso a Arquivos

fseek

- Para se fazer procuras e acessos randômicos em arquivos usa-se a função fseek(). Esta move a posição corrente de leitura ou escrita no arquivo de um valor especificado, a partir de um ponto especificado. Seu protótipo é:

  ```
  int fseek (FILE *fp, long numbytes, int origem);
  ```

Outros Comandos de Acesso a Arquivos

fseek

O parâmetro origem determina a partir de onde os numbytes de movimentação serão contados. Os valores possíveis são definidos por macros em <stdio.h> e são:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente no arquivo
SEEK_END	2	Fim do arquivo

Tendo-se definido a partir de onde irá se contar, numbytes determina quantos bytes de deslocamento serão dados na posição atual.

Outros Comandos de Acesso a Arquivos

rewind

- Retorna a posição corrente do arquivo para o início. Protótipo:

```
void rewind (FILE *fp);
```

remove

- Apaga um arquivo especificado. Protótipo:

```
int remove (char *nome_do_arquivo);
```

Outros Comandos de Acesso a Arquivos

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main()
{
    FILE *p;
    char str[30], frase[] = "Este e um arquivo chamado: ";
    char resposta[80];
    int i;
```

Outros Comandos de Acesso a Arquivos

```
/* Le um nome para o arquivo a ser aberto: */
printf("\n\n Entre com um nome para o arquivo:\n");

/* Usa fgets como se fosse gets */
fgets(str, 29, stdin);

/* Elimina o \n da string lida */
for(i = 0; str[i]; i++)
   if(str[i] == '\n')
      str[i] = 0;
if (!(p = fopen(str,"w")))
          /* Caso ocorra algum erro na abertura do arquivo..*/
{         /* o programa aborta automaticamente */
          printf("Erro! Impossivel abrir o arquivo!\n");
          exit(1);
}
```

Outros Comandos de Acesso a Arquivos

```
/* Se nao houve erro, imprime no arquivo, e o fecha ...*/
fputs(frase, p);
fputs(str,p);
fclose(p);

/* abre novamente e le */
p = fopen(str,"r");
fgets(resposta, 79, p);
printf("\n\n%s\n", resposta);
fclose(p); /* Fecha o arquivo */
remove(str); /* Apaga o arquivo */
return(0);
}
```

Exercício 1

- Considere a estrutura:

```
struct aluno{
    char cod[3];
    char nome[20];
    char idade[3];
    };
```

- Faça um programa para ler os dados de 3 alunos e gravar em um arquivo chamado alunos.txt. Após gravar, ler o arquivo e imprimir tudo o que for lido.

Exercício 2

- Faça um programa para criar um arquivo chamado “produtos.dat”, onde cada registro será composto pelos seguintes campos: codigo, descrição e preço. Faça uma função para consultar todos os produtos que possuem preço superior a R$500,00.