unipampa
Universidade Federal do Pampa

ENGENHARIA DE COMPUTAÇÃO

Laboratório de Programação I

Assunto de Hoje:
Mais Alguns Comandos Úteis

lucianobrum@unipampa.edu.br

Comandos Adicionais

- Além de permitir manipular arquivos, a linguagem C também permite apagá-lo do disco. Isso pode ser feito utilizando a função remove:

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

- Como retorno temos um valor inteiro, o qual será igual a 0 se o arquivo for excluído com sucesso.

Exemplo

```
int main(){
    int status;
    status = remove("ArqGrav.txt");
    if(status != 0){
        printf("Erro na remocao do arquivo.\n");
        system("pause");
        exit(1);
    }else
        printf("Arquivo removido com sucesso.\n");

    return 0;
}
```

Comandos Adicionais

- Para realizar a leitura/escrita de um registro específico, deve-se posicionar o ponteiro de forma adequada;

- Comando para posicionamento do "ponteiro de leitura/escrita" em arquivos binários:

```
fseek(FILE *fp, long int pos_em_bytes, int modo);
```

Comandos Adicionais

- O **fp** é aquele de quando abrimos o arquivo (Variável FILE *).

- O **pos_em_bytes** é a posição, em bytes, para onde moveremos o ponteiro interno do arquivo. É um valor long int, então ao declarar alguma variável para isso devemos usar o tipo long, e ao usar números diretamente é recomendável usar o molde (long), para caso do valor passar o tamanho máximo suportado para uma variável int.

Comandos Adicionais

- O modo indica como queremos mover o ponteiro. São três modos:
- **SEEK_SET (constante de valor 0)** - movimenta para a posição indicada (começando a contar do zero, que representa o primeiro byte do arquivo).
- **SEEK_END (constante de valor 1)** - movimenta para a posição indicada, começando a contar do final do arquivo. Neste caso, o zero representa a posição imediatamente posterior ao último byte do arquivo.
- **SEEK_CUR (constante de valor 2)** - movimenta a partir da posição atual. Neste caso podemos colocar números negativos, que significa que queremos retroceder com o ponteiro do arquivo. Qualquer operação de leitura ou gravação em um arquivo move o ponteiro interno deste arquivo o mesmo números de bytes da operação.
- Quando abrimos um arquivo no modo append (a), o ponteiro começa na posição zero, em relação ao final do arquivo. Seria como fazer fseek usando o SEEK_END.

Exemplo

```
int result;
FILE *fp;
fp = fopen("arquivo.bin", "r+b");
if(fp!=0){
printf("Erro na abertura do arquivo\n");
exit(fp);}
result = fseek(fp,0,SEEK_SET);
//equivale ao rewind(fp);
if(result!=0)
printf("Erro no posicionamento!\n");
```

Comandos Adicionais

- Lendo a posição do ponteiro de leitura/gravação do arquivo :
- O comando *ftell* informa a posição atual do ponteiro interno do arquivo. Exemplo:
- long posicao;
- posicao = ftell(ponteiro_arq)/sizeof(struct cliente);
- printf ("O ponteiro interno do arquivo esta' apontando para o %ldº registro.\n", posicao + 1);

Comandos Adicionais

- Este comando pode ser útil também para ver qual é o tamanho de um arquivo.
- Para isso, posicione o ponteiro do arquivo no final do arquivo (usando SEEK_END), e depois leia a posição.
- O número retornado é o tamanho do arquivo, em bytes.

Comandos Adicionais

- sscanf = função para leitura de valores formatados de uma string;
- Exemplo:
- sscanf(char *str, char *frmt, var1, ...)
- sscanf(calculo, "v1 %f op %c v2 %f", &operando1, &operacao, &operando2);

Comandos Adicionais

- **Strtok:** devolve um ponteiro para a próxima palavra na string apontada por **endereçoStrOrigem**.
- Os caracteres que formam a string apontada por **endereçoStrDelimitador** são os delimitadores que terminam a palavra. Um ponteiro nulo é devolvido quando não há mais palavras na string.

- char *strtok(char * endereçoStrOrigem, char * endereçoStrDelimitador);

Comandos Adicionais

- Na primeira chamada à função strtok, o endereçoStrOrigem é realmente utilizado na chamada. Nas chamadas seguintes deve-se usar um ponteiro nulo como primeiro argumento.

- **Exemplo**.

unipampa
Universidade Federal do Pampa

Engenharia de Computação

Dúvidas ?