# Capítulo 22. Exemplos: MÓDULOS arquivo e plota; e arquivos 2

**27 Nov 2024**

## OBJETIVOS DO CAPÍTULO

- Ler dados de arquivo sem tamanho conhecido
- Generalizar o nome do arquivo de dados
- Usar um módulo de manipulação de arquivos
- Usar um módulo de geração de gráficos
- Comando novo do FORTRAN: DO WHILE
- Uso avançado do comando READ

## 22.1 programa DO_WHILE_2

1) Objetivos do programa:
   a) Ler dados de arquivo sem tamanho conhecido, empregando os comandos READ e DO WHILE
   b) Usar nome do arquivo de dados definido pelo usuário
2) No Fortran, **criar um projeto** com o nome **do_while_2**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **calibra_2p0_fonte.f90**
4) No Fortran, **copiar** exatamente o texto mostrado na **Tabela 22.1**.
5) Na pasta do projeto, também deverão ser incluídos os seguintes arquivos do exemplo:
   - **calibra_2p0_dados_geral.txt** (arquivo de dados de nome fixo)
   - **m1g_dados.txt** (arquivo de dados com nome definido pelo usuário)
   - **calibra_2p0.gnu** (arquivo de comandos para o aplicativo Wgnuplot)
   - **m1g.txt** (arquivo com os dados a serem plotados no gráfico)
   - **Wgnuplot.exe** (aplicativo para fazer gráfico)
6) Todos arquivos acima estão disponíveis **AQUI**.
7) Comentários sobre o programa:
   a) Este programa lê dados "x e y" de um arquivo de tamanho desconhecido, isto é, não se sabe a priori qual é a quantidade de dados a ser lida. Depois, faz um gráfico destes dados lidos.
   b) Este programa exemplifica um procedimento que pode ser utilizado para o usuário entrar com nomes diferentes do arquivo de dados: usa-se um primeiro arquivo de dados de nome fixo (`calibra_2p0_dados_geral.txt`), através do qual o usuário entra com o nome do seu arquivo de dados (variável dados).
   c) Para ler a quantidade desconhecida de dados, é usado o novo comando DO WHILE, que é um ciclo executado enquanto uma condição estiver sendo satisfeita [`do while ( flag == 0 )`].
8) Executar **Build, Compile** para compilar o programa

Tabela 22.1 Calibra_2p0_fonte.f90.

```
Program Calibra_2p0

use portlib
implicit none
real*8  :: t, E, to
integer :: ver, flag, io, i
character*50 :: original, dados
character*70 :: titulo

! Mostra o conteúdo do arquivo de dados com nome fixo
ver = system("Notepad calibra_2p0_dados_geral.txt")

! Lê o nome do arquivo específico e genérico
open(8,file="calibra_2p0_dados_geral.txt")
read(8,*) dados
close(8)

! Mostra o conteúdo do arquivo específico e genérico
ver = system('notepad '//dados)

! Lê os dados do arquivo específico e genérico
open(8,file=dados)
read(8,*) original
read(8,*) titulo
close(8)

! Mostra o conteúdo do arquivo de dados
ver = system('notepad '//original)

! *** CRIA O ARQUIVO DE SAÍDA PARA GRÁFICO ***
open(20,file="grafico_dados.txt")

! lê os dados do arquivo sem saber o seu comprimento
open(8,file=trim(adjustl(original)))
flag  =  0
i = 0
do while ( flag == 0 )
   i = i + 1
   read(8,*,iostat=io) t, E
   if ( i == 1 ) to = t
   t = t - to
   if ( io >= 0 ) write(20,21) t, E
   21 format ( 2 (1pe15.6) )
   if ( io < 0 ) flag = 1
end do
close(8)

close(20)

! *** gera o gráfico da curva de força ***
open(18,file="calibra_2p0.gnu")
do i = 1, 6
   read(18,*)
end do
write(18,31) trim(adjustl(titulo))
31 format("set title '", a, "'")
write(18,*) 'plot "grafico_dados.txt" notitle'
close(18)

ver = system("Wgnuplot calibra_2p0.gnu")

end program Calibra_2p0
```

9) Gerar o programa-executável fazendo **Build, Build**.
10) Executar o programa através de **Build, Execute com os dados que estão nos arquivos deste exemplo.** O resultado final deverá ser o gráfico mostrado na Figura 22.1.
11) **Analisar o programa-fonte**.

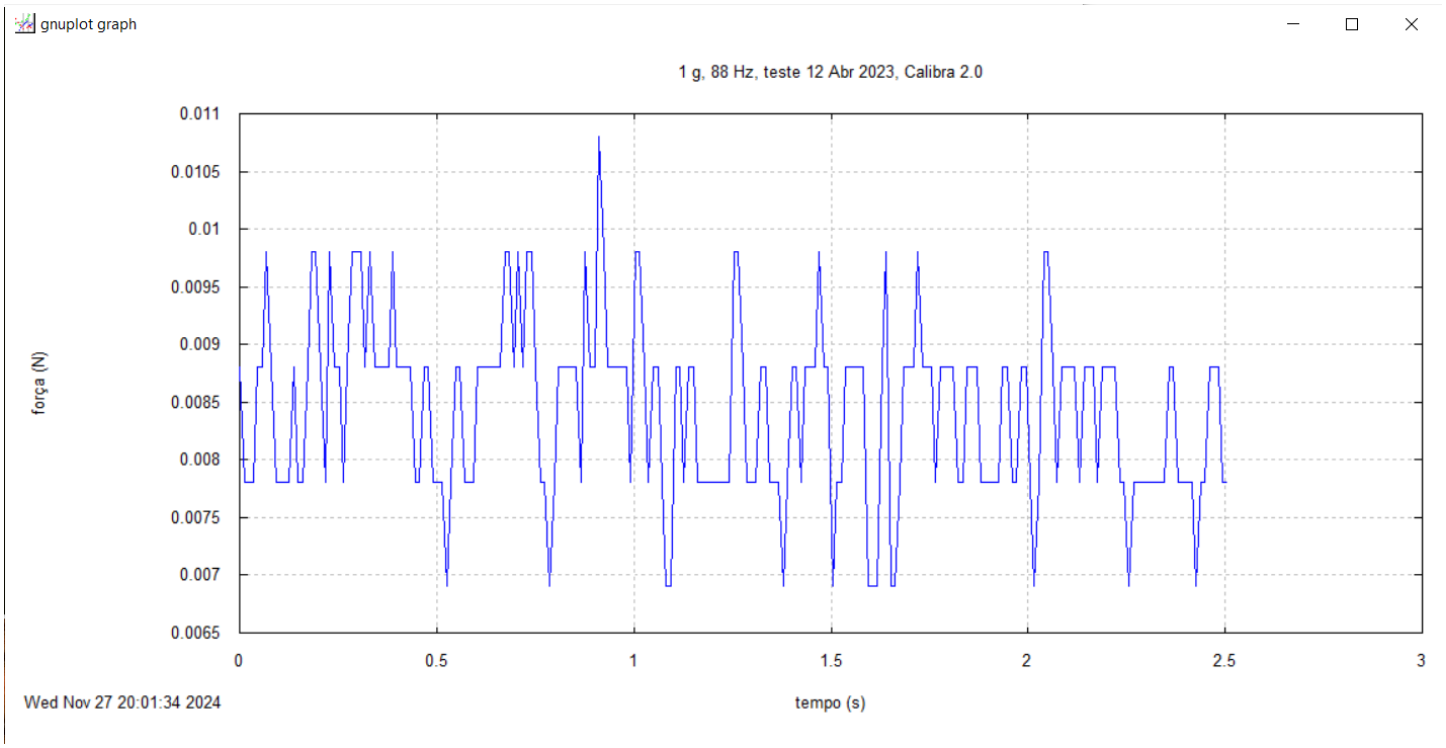


Figura 22.1 Gráfico resultante da execução do projeto do_while_2.

## 22.2 projeto ARQUIVO2

1) Objetivo do programa: usar um módulo de manipulação de arquivos
2) No Fortran, **criar um projeto** com o nome **arquivo2**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **main_arquivo.f90**
4) No Fortran, **copiar** exatamente o texto mostrado na **Tabela 22.2**.
5) Na pasta do projeto, também deverão ser incluídos os seguintes arquivos do exemplo:
   - **inicio.arquivo.txt** (arquivo de inicialização do módulo ARQUIVO)
   - **arquivo.ent** (arquivo de dados do módulo ARQUIVO)
   - **arquivo.f90** (arquivo com o programa-fonte do módulo ARQUIVO)
6) Todos arquivos acima estão disponíveis **AQUI**.
7) Comentários sobre o programa:
   a) O módulo ARQUIVO tem 24 sub-rotinas. Nove delas são usadas no exemplo do programa-fonte main_arquivo.f90.
   b) Este módulo é dedicado à manipulação de arquivos: criar arquivo, criar pasta, copiar arquivo, acessar o conteúdo de um arquivo, cronometrar o tempo de CPU etc.

Tabela 22.2 Main_arquivo.f90.

```
program modulo_arquivo

use arquivo

integer         :: unit = 10 ! unidade do arquivo
character*(50) :: nome = "inicio.arquivo.txt" ! nome do arquivo de dados de configuração do módulo
character*12    :: dia ! data
real*8          :: tcpu_i ! zero do cronômetro
real*8          :: tcpu_f ! tempo de processamento
character*(50) :: nome_saida = "out.txt" ! nome do arquivo de saída
integer         :: ver ! acessa dos com o comando system

call ARQUIVO_cronometro ( tcpu_i )

call ARQUIVO_notepad_abre_dado ( nome )

call ARQUIVO_le_inicio ( unit, nome )

call ARQUIVO_cria_arquivo ( unit, nome_saida )

call ARQUIVO_escreve_data ( unit )

call ARQUIVO_escreve_hora ( unit )

call ARQUIVO_passa_data ( dia )
write(unit,*) "data de hoje: ", dia

call ARQUIVO_cronometro ( tcpu_f )

call ARQUIVO_escreve_tempo_cpu ( tcpu_i, tcpu_f, unit )

close(unit)

call ARQUIVO_notepad_abre_saida ( unit, nome_saida )

end program modulo_arquivo
```

8) Executar **Build, Compile** para compilar o programa-fonte **arquivo.f90** e, depois, o **main_arquivo.f90**.

9) Gerar o programa-executável fazendo **Build, Build**.

10) Executar o programa através de **Build, Execute com os dados que estão nos arquivos deste exemplo.** O resultado final deverá ser o arquivo mostrado na Figura 22.2 e os comandos da Figura 22.3.

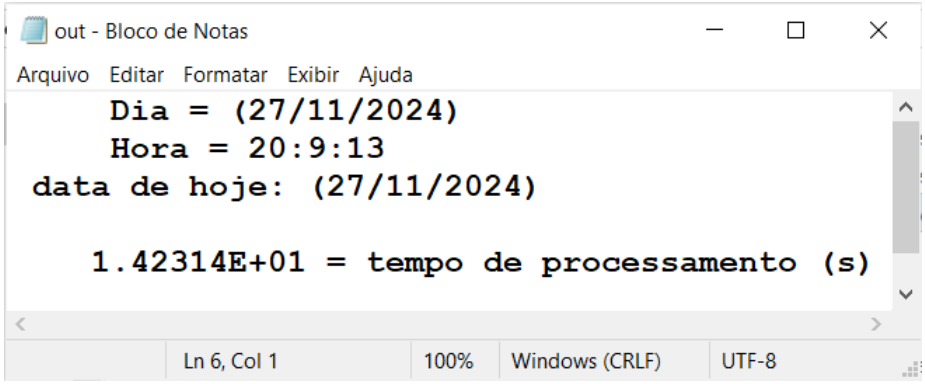


Figura 22.2 Arquivo de saída do projeto Arquivo2.

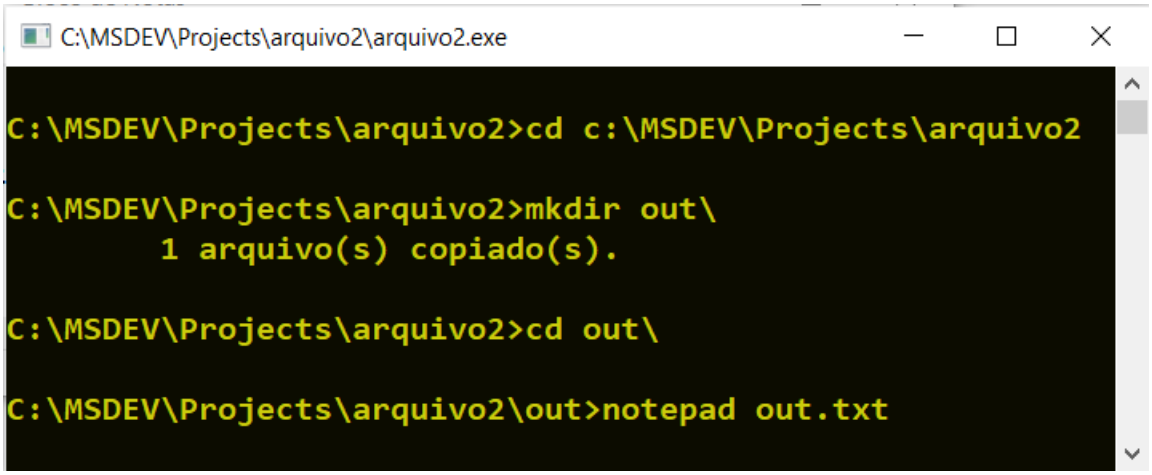


Figura 22.3 Comandos do DOS executados pelo projeto Arquivo2.

11) **Analisar os arquivos deste projeto**.

## 22.3 projeto PLOTA

1) Objetivo do programa: usar um módulo de geração de gráficos
2) No Fortran, **criar um projeto** com o nome **PLOTA**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **main_plota.f90**
4) No Fortran, **copiar** exatamente o texto mostrado na **Tabela 22.3**.
5) Na pasta do projeto, também deverão ser incluídos os seguintes arquivos do exemplo:
   - **inicio.arquivo.txt** (arquivo de inicialização do módulo ARQUIVO)
   - **arquivo.ent** (arquivo de dados do módulo ARQUIVO)
   - **arquivo.f90** (arquivo com o programa-fonte do módulo ARQUIVO)
   - **inicio.plota.ent** (arquivo de inicialização do módulo PLOTA)
   - **plota.ent** (arquivo de dados do módulo PLOTA)
   - **plota.f90** (arquivo com o programa-fonte do módulo PLOTA)
   - **cos_000.Mach.dat** (arquivo com os dados a serem plotados no gráfico)
   - **Mach.grafico.ent** (arquivo com opções para gerar o gráfico)
   - **Wgnuplot.exe** (aplicativo para fazer gráfico)
6) Todos arquivos acima estão disponíveis **AQUI**.
7) Comentários sobre o programa:
   a) O módulo ARQUIVO é o mesmo da seção 22.2, anterior.
   b) O módulo PLOTA tem 7 sub-rotinas. Seis delas são usadas no exemplo do programa-fonte main_plota.f90.
   c) Este módulo é dedicado à geração de gráficos, que podem ser moldados, em vários aspectos, de acordo com o desejado pelo usuário.

Tabela 22.3 Main_plota.f90.

```
program modulo_plota

use plota

integer         :: unit = 10 ! unidade do arquivo
character*(50) :: nome_arq  = "inicio.arquivo.txt" ! nome do arquivo de dados de
configuração do módulo ARQUIVO
character*(50) :: nome_plot = "inicio.plota.ent" ! nome do arquivo de dados de
configuração do módulo PLOTA
character*(50) :: nome_caso ! nome do caso a analisar

call ARQUIVO_notepad_abre_dado ( nome_arq )

call ARQUIVO_le_inicio ( unit, nome_arq )

call ARQUIVO_passa_caso ( nome_caso )

call PLOTA_copia_Wgnuplot

call ARQUIVO_notepad_abre_dado ( nome_plot )

call PLOTA_le_inicio ( unit, nome_plot )

call PLOTA_gera_nomes ( unit, '.Mach.dat' )

call PLOTA_le_dados_gnu (  unit, 'Mach.grafico.ent' )

call PLOTA_gera_gnu ( unit, trim(nome_caso)//'.Mach.grafico.gnu' )

call PLOTA_executa_Wgnuplot ( unit, trim(nome_caso)//'.Mach.grafico.gnu' )

end program modulo_plota
```

8) Executar **Build, Compile** para compilar os programas-fonte (nesta ordem): **arquivo.f90**, **plota.f90** e **main_plota.f90**.
9) Gerar o programa-executável fazendo **Build, Build**.
10) Executar o programa através de **Build, Execute com os dados que estão nos arquivos deste exemplo.** O resultado final deverá ser o gráfico mostrado na Figura 22.4 e os comandos da Figura 22.5.
11) **Analisar os arquivos deste projeto**.

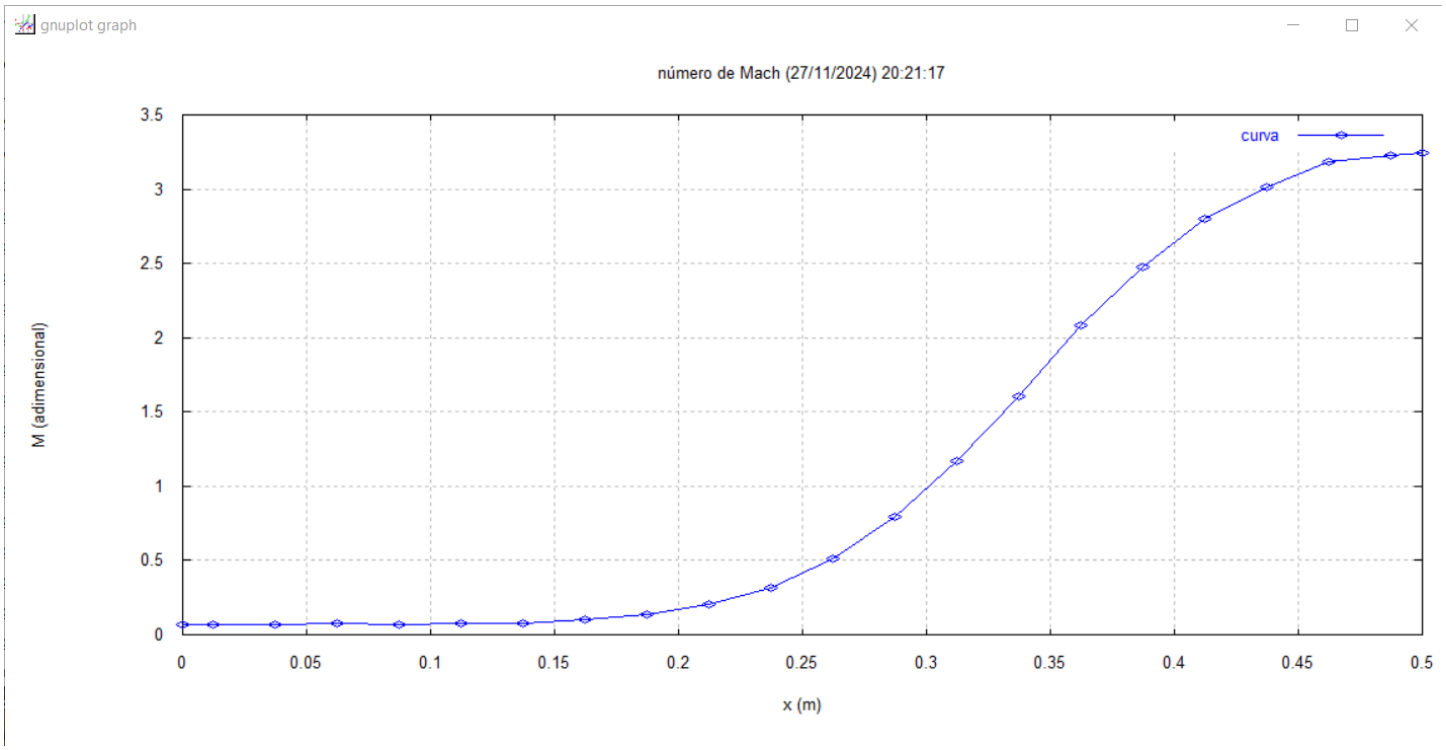


Figura 22.4 Gráfico resultante da execução do projeto Plota.

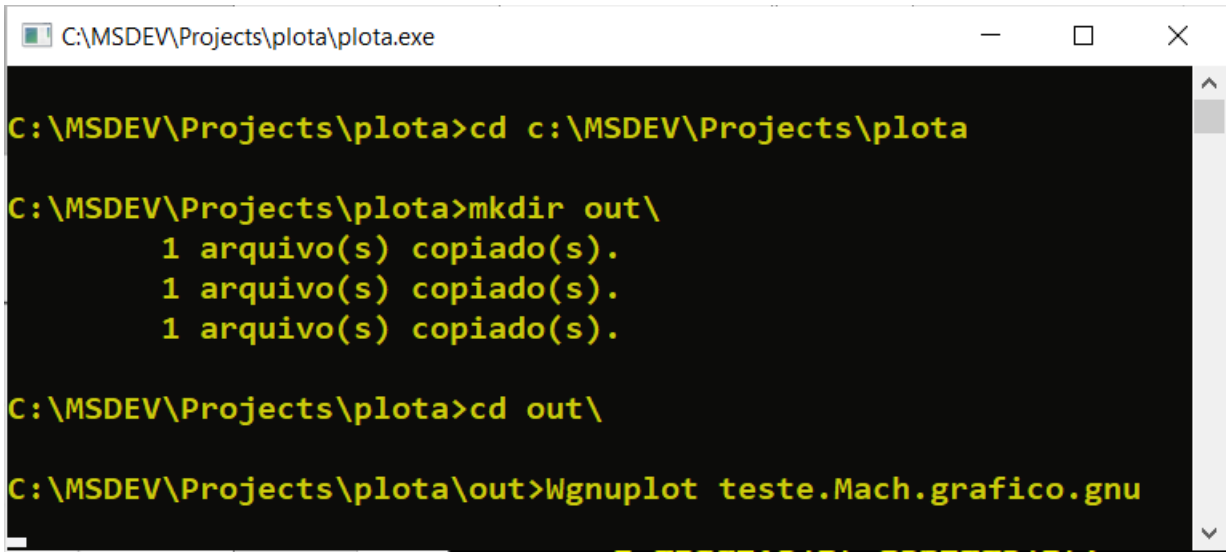


Figura 22.5 Comandos do DOS executados pelo projeto Plota.