CCET

UNIRIO / CCET - Ensino e Pesquisa - Produzir e disseminar conhecimento

UNIRIO

Guia Linux

GCC

gcc [opções]

Descrição

Este aplicativo compila programas escritos nas linguagens C e C++.

Algumas opções do comando

- **-c** : compila e converte para a linguagem de máquina sem linkar (não cria o programa executável).
- **-D macro** : define nome de macro a ser usada dentro do programa (podemos também definir uma macro dentro do programa C através da inclusão da linha "#define macro"). O uso de macros permite selecionar quais partes do código C devem ser compiladas. Por exemplo,

```
#ifdef teste_macro
conjunto de instruções 1
#else
conjunto de instruções 2
#endif
```

  O compilador usará o conjunto de instruções 1 se a macro *teste_macro* tiver sido definida, senão usará o conjunto de instruções 2.

- **-E** : faz apenas o pré-processamento do código-fonte. Não compila.
- **-I diretório** ou **-Idiretório** : adiciona diretório a lista dos diretórios pesquisados na busca por arquivos definidos por um comando *include*.
- **-l biblioteca** ou **-lbiblioteca**: adiciona biblioteca durante a linkedição.
- **-L diretório** ou **-Ldiretório**: adiciona diretório a lista dos diretórios pesquisados na busca por arquivos definidos como bibliotecas compartilhadas. Use -L. para incluir o diretório corrente.
- **-o arquivo** : nome do arquivo de saída (o nome padrão para o arquivo executável é *a.out*).
- **-S** : compila o arquivo pré-processado e cria o código *assembly*.

- **-save-temps** : não deleta os arquivos temporários gerados na compilação.
- **-shared** : produz um objeto compartilhado que pode então ser linkeditado com outros objetos para formar um executável.
- **-v** : informa os comandos usados pelo GCC na compilação do arquivo.
- **-w** : omite todas as mensagens de advertência da compilação.
- **-Wall** : exibe todas as mensagens de advertência da compilação.

Exemplos

O comando

```
gcc teste.c -o teste -lm
```

compila o arquivo *teste.c* e cria, caso não haja erro de compilação, o arquivo executável *teste*. Neste caso, o comando usa a biblioteca de matemática na linkedição.

Na realidade, o GCC executa 4 passos para obter o executável de um programa. É possível executar cada um desses passos separadamente. Vejamos como isto pode ser feito para o arquivo *teste.c*.

- **Pré-processamento –** processa cabeçalhos e macros.

```
gcc -E teste.c -o teste.i
```

- **Compilação –** compila e converte o arquivo *teste.i* para a linguagem *assembly*.

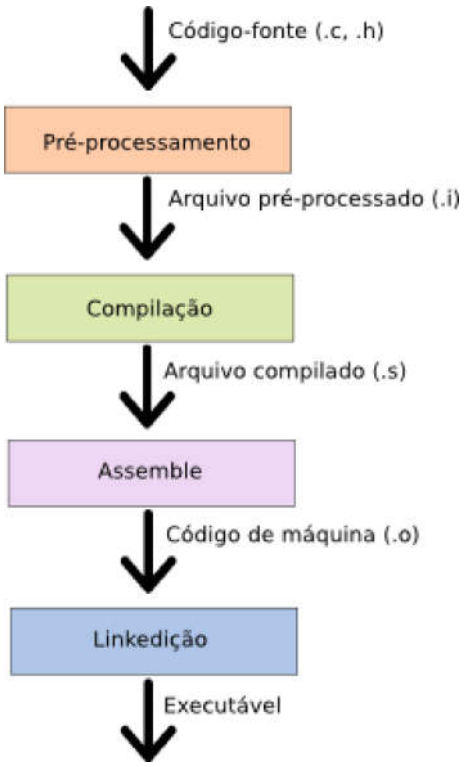
```
gcc -S teste.i -o teste.s
```

- **Assemble –** cria o arquivo objeto (código de máquina).

```
gcc -c teste.s -o teste.o
```

- **Linkedição –** cria o executável adicionando as bibliotecas.

```
gcc teste.o -o teste -lm
```



Os comandos acima dão a falsa ideia que o GCC faz tudo sozinho. Na realidade, o GCC usa programas externos para executar estas tarefas. Para descobrir quais os programas chamados pelo GCC na criação do arquivo *teste*, digite

```
gcc -v -save-temps teste.c -o teste
```

Note que os arquivos temporários não foram deletados. Isto se deve à opção de depuração *save-temps* usada no exemplo.

Observações

- A **biblioteca padrão de C** no Linux é a **glibc**.
- **GCC** é um compilador do **projeto GNU** para C e C++. Mas já existem interfaces disponíveis no GCC para outras linguagens como Java e Fortran.

Sumário | Topo