

**GCC
2023
NANU**

GCC, a Coleção de Compiladores GNU

A GNU Compiler Collection inclui front-ends para C, C++, Objective-C, Fortran, Ada, Go e D, bem como bibliotecas para essas linguagens (libstdc++,...). GCC foi originalmente escrito como o compilador para o sistema operacional GNU. O sistema GNU foi desenvolvido para ser um software 100% livre, livre no sentido de [respeitar a liberdade do usuário](#).

Nós nos esforçamos para fornecer [lançamentos](#) regulares e de alta qualidade, que queremos que funcionem bem em uma variedade de alvos nativos e cruzados (incluindo GNU/Linux), e encorajamos todos a [contribuir](#) com mudanças ou ajudar a [testar](#) o GCC. Nossas fontes estão prontamente e gratuitamente disponíveis via [Git](#) e [snapshots](#) semanais.

As principais decisões sobre o GCC são tomadas pelo [comitê gestor](#), guiadas pela [declaração de missão](#).

Notícias

GCC BPF no Compiler Explorer [2022-12-23]

O suporte para uma compilação noturna do compilador bpf-unknown-none-gcc foi contribuído para o Compiler Explorer (também conhecido como godbolt.org) por Marc Poulhiès

GNU Tools Caldeirão 2022 [2022-09-02]

Praga, República Tcheca e online, 16 a 18 de setembro de 2022

GCC 12.2 lançado [2022-08-19]

GCC 10.4 lançado [2022-06-28]

GCC 9.5 lançado [2022-05-27]

GCC 12.1 lançado [2022-05-06]

GCC 11.3 lançado [2022-04-21]

[Notícias mais antigas](#) | Mais notícias? Avise gerald@pfeifer.com!

Versões suportadas

GCC 12.2 (alterações)

Status: 2022-08-19 (apenas correções de regressão e documentos).

[Regressões graves](#). [Todas as regressões](#).

GCC 11.3 (alterações)

Status: 21/04/2022 (apenas correções de regressão e documentos).

[Regressões graves](#). [Todas as regressões](#).

GCC 10.4 (alterações)

Status: 2022-10-19 (apenas correções de regressão e documentos).

[Regressões graves](#). [Todas as regressões](#).

Desenvolvimento: GCC 13.0 (critérios de lançamento, mudanças)

Status: 2023-01-16 (apenas correções de regressão e documentos).

[Regressões graves](#). [Todas as regressões](#).

Pesquise em nosso site

 Correspondência: Todas as palavras Ordenar por: O mais novo

Há também um [formulário de pesquisa detalhado](#).

Receba nossos anúncios



Receba o GCC

o

Instalações

[Listas de discussão](#)

[Colaboradores](#)



@gnutools

[Make A Donation](#)

Documentação

[Instalação](#)

[Manual de Plataformas](#)

[FAQ Wiki Pointers](#)

Download

[Binários de espelhos](#)

Fontes

[Git](#)

[... acesso de gravação](#)

[Rsync](#)

Desenvolvimento

[Plano e Cronograma](#)

[Contribuição](#)

[Por que contribuir?](#)

[Projetos abertos](#)

Instalando o GCC: Edifício

Agora que o GCC está configurado, você está pronto para construir o compilador e as bibliotecas de tempo de execução.

Alguns comandos executados ao fazer o compilador podem falhar (retornar um status diferente de zero) e ser ignorados pelo `make`. Essas falhas, que geralmente ocorrem devido a arquivos que não foram encontrados, são esperadas e podem ser ignoradas com segurança.

É normal ter avisos do compilador ao compilar determinados arquivos. A menos que você seja um desenvolvedor GCC, geralmente pode ignorar esses avisos, a menos que causem falha na compilação. Os desenvolvedores devem tentar corrigir quaisquer avisos encontrados, no entanto, eles podem continuar temporariamente após os avisos como erros, especificando o sinalizador de configuração `--disable-werror`.

Em certos sistemas antigos, a definição de determinadas variáveis de ambiente, como as que `cc` podem interferir no funcionamento do `make`.

Se você encontrar erros aparentemente estranhos ao tentar compilar o compilador em um diretório diferente do diretório de origem, pode ser porque você configurou anteriormente o compilador no diretório de origem. Certifique-se de ter feito todos os preparativos necessários.

Se você construir o GCC em um sistema BSD usando um diretório armazenado em um sistema de arquivos System V antigo, poderão ocorrer problemas na execução `fixincl` se o sistema de arquivos System V não suportar links simbólicos. Esses problemas resultam em uma falha na correção da declaração `size_t` de `sys/types.h`. Se você achar que `size_t` é um tipo assinado e que ocorrem incompatibilidades de tipo, essa pode ser a causa.

A solução é não usar esse diretório para construir o GCC.

Da mesma forma, ao compilar a partir do repositório de origem ou instantâneos, ou se você modificar `*.uarquivos`, você precisa do gerador de analisador léxico Flex instalado. Se você não modificar `*.uarquivos`, as versões contêm os arquivos gerados pelo Flex e você não precisa do Flex instalado para criá-los. Ainda existe um analisador léxico baseado em Flex (parte do maquinário de construção, não do próprio GCC) que é usado mesmo se você construir apenas o front-end C.

Ao compilar a partir do repositório de origem ou instantâneos, ou se você modificar a documentação do Texinfo, precisará da versão 4.7 ou posterior do Texinfo instalada se desejar que a documentação do Info seja regenerada. Os lançamentos contêm documentação de informações pré-criada para a documentação não modificada no lançamento.

Construindo um compilador nativo

Para uma compilação nativa, a configuração padrão é executar uma inicialização de 3 estágios do compilador quando `'fazer'` é invocado. Isso criará todo o sistema GCC e garantirá que ele seja compilado corretamente. Pode ser desabilitado com o `--disable-bootstrap` parâmetro para `'configurar'`, mas a inicialização é sugerida porque o compilador será testado de forma mais completa e também poderá ter

A seguir: [Casos de uso para o GNU Build System](#), Acima: [Uma introdução às ferramentas automáticas](#) [[Conteúdo](#)][[Índice](#)]

2.1 Apresentando o GNU Build System

É uma verdade universalmente reconhecida que, como um desenvolvedor de posse de um novo pacote, você deve estar precisando de um sistema de construção.

No mundo Unix, esse sistema de compilação é tradicionalmente obtido usando o comando `make` (consulte [Visão geral](#) no Manual do GNU Make). Você expressa a receita para construir seu pacote em um `Makefile`. Este arquivo é um conjunto de regras para construir os arquivos no pacote. Por exemplo o `programa.prog` pode ser criado executando o vinculador nos arquivos `principal.o`, `foo.o`, e `bar.o`; o arquivo `principal.o` pode ser construído executando o compilador em `main.c`; etc. Cada vez que `make` é executado, ele lê `Makefile`, verifica a existência e o tempo de modificação dos arquivos mencionados, decide quais arquivos precisam ser compilados (ou reconstruídos) e executa os comandos associados.

Quando um pacote precisa ser construído em uma plataforma diferente daquela em que foi desenvolvido, seu `Makefile` geralmente precisa ser ajustado. Por exemplo, o compilador pode ter outro nome ou exigir mais opções. Em 1991, David J. MacKenzie se cansou de customizar `Makefile` para as 20 plataformas com as quais ele teve que lidar. Em vez disso, ele criou um pequeno script de shell chamado `configure` para ajustar automaticamente o `Makefile` (veja [Gênesis](#) no Manual do Autoconf). Compilar seu pacote agora era tão simples quanto executar `./configure && make`.

Hoje esse processo foi padronizado no projeto GNU. Os Padrões de Codificação GNU (consulte [O Processo de Lançamento](#) em Os Padrões de Codificação GNU) explica como cada pacote do projeto GNU deve ter um `configure` script e a interface mínima que ele deve ter. O `Makefile` também deve seguir algumas convenções estabelecidas. O resultado? Um sistema de compilação unificado que torna todos os pacotes quase

Tendo essas paginas como referencias
Busque todas as informações nelas
Não existe caminho facil.
Leia a doc e busque tutoriais
Aqui começa o caminho
GNU.

Documentação on-line do GCC

Últimos lançamentos

Estes são manuais para os últimos lançamentos completos.

▼ Manuais do GCC 12.2:

- [GCC 12.2 Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 GNU Fortran Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 CPP Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 GNAT Reference Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 GNAT User's Guide](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 Standard C++ Library Manual](#) (também em [PDF](#) ou [XML](#) ou um [tarball HTML](#))
- [GCC 12.2 Standard C++ Library Reference Manual](#) (também em [PDF](#) ou [XML GPL](#) ou [XML GFDL](#) ou um [tarball HTML](#))
- [GCCGO 12.2 Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 GNU Offloading and Multi Processing Runtime Library Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 12.2 Quad-Precision Math Library Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [Biblioteca GCC 12.2 JIT](#)
- [Fontes Texinfo de todos os manuais GCC 12.2](#)

▼ Manuais GCC 11.3:

- [GCC 11.3 Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 GNU Fortran Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 CPP Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 GNAT Reference Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 GNAT User's Guide](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 Standard C++ Library Manual](#) (também em [PDF](#) ou [XML](#) ou um [tarball HTML](#))
- [GCC 11.3 Standard C++ Library Reference Manual](#) (também em [PDF](#) ou [XML GPL](#) ou [XML GFDL](#) ou um [tarball HTML](#))
- [GCCGO 11.3 Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 GNU Offloading and Multi Processing Runtime Library Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))
- [GCC 11.3 Quad-Precision Math Library Manual](#) (também em [PDF](#) ou [PostScript](#) ou um [tarball HTML](#))

Instalando o GCC: Edifício

Agora que o GCC está configurado, você está pronto para construir o compilador e as bibliotecas de tempo de execução.

Alguns comandos executados ao fazer o compilador podem falhar (retornar um status diferente de zero) e ser ignorados pelo `make`. Essas falhas, que geralmente ocorrem devido a arquivos que não foram encontrados, são esperadas e podem ser ignoradas com segurança.

É normal ter avisos do compilador ao compilar determinados arquivos. A menos que você seja um desenvolvedor GCC, geralmente pode ignorar esses avisos, a menos que causem falha na compilação. Os desenvolvedores devem tentar corrigir quaisquer avisos encontrados, no entanto, eles podem continuar temporariamente após os avisos como erros, especificando o sinalizador de configuração `--disable-werror`.

Em certos sistemas antigos, a definição de determinadas variáveis de ambiente, como as que `cc` podem interferir no funcionamento do `make`.

Se você encontrar erros aparentemente estranhos ao tentar compilar o compilador em um diretório diferente do diretório de origem, pode ser porque você configurou anteriormente o compilador no diretório de origem. Certifique-se de ter feito todos os preparativos necessários.

Se você construir o GCC em um sistema BSD usando um diretório armazenado em um sistema de arquivos System V antigo, poderão ocorrer problemas na execução `fixincl` se o sistema de arquivos System V não suportar links simbólicos. Esses problemas resultam em uma falha na correção da declaração `size_t` de `sys/types.h`. Se você achar que `size_t` é um tipo assinado e que ocorrem incompatibilidades de tipo, essa pode ser a causa.

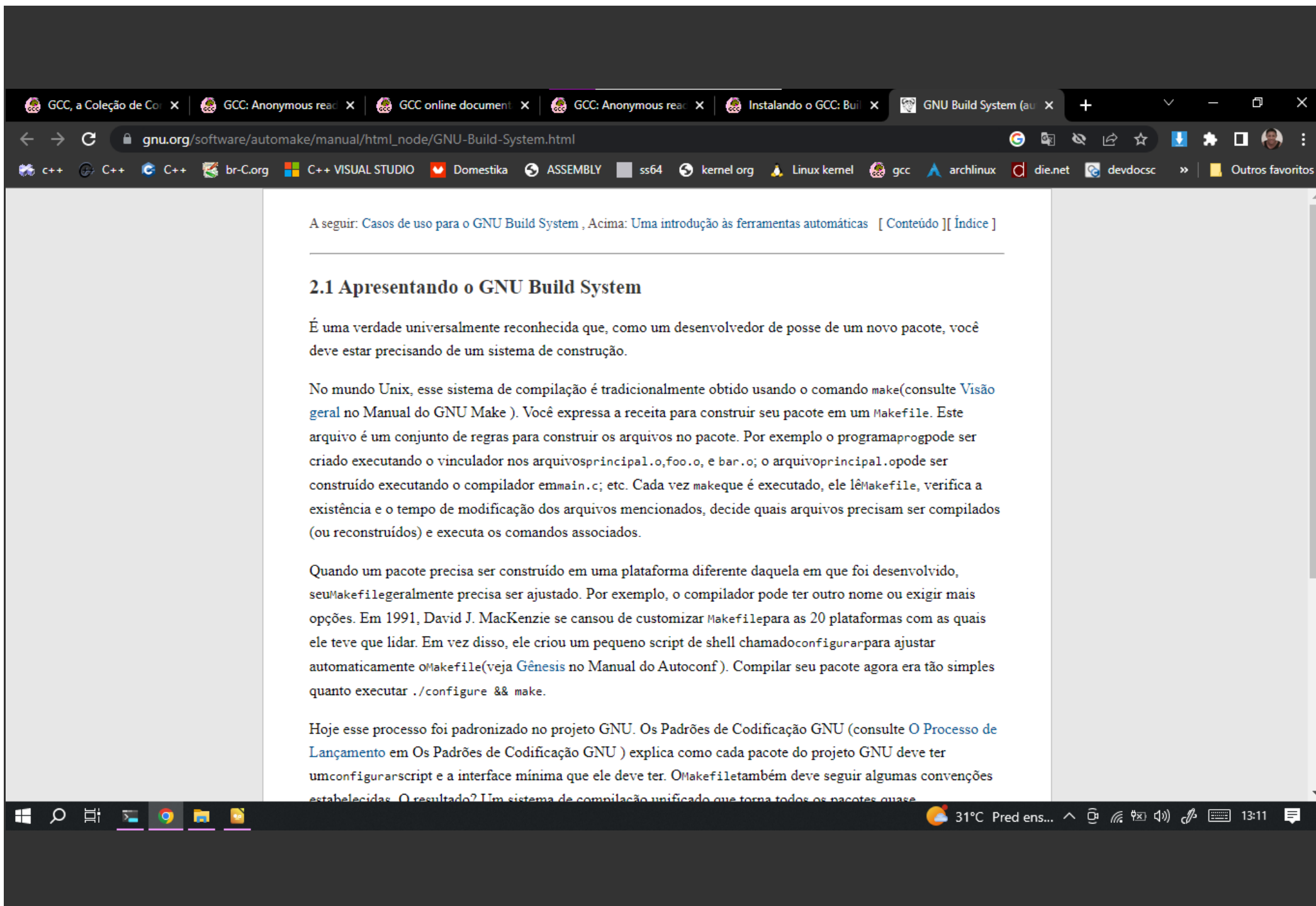
A solução é não usar esse diretório para construir o GCC.

Da mesma forma, ao compilar a partir do repositório de origem ou instantâneos, ou se você modificar `*.euarquivos`, você precisa do gerador de analisador léxico Flex instalado. Se você não modificar `*.euarquivos`, as versões contêm os arquivos gerados pelo Flex e você não precisa do Flex instalado para criá-los. Ainda existe um analisador léxico baseado em Flex (parte do maquinário de construção, não do próprio GCC) que é usado mesmo se você construir apenas o front-end C.

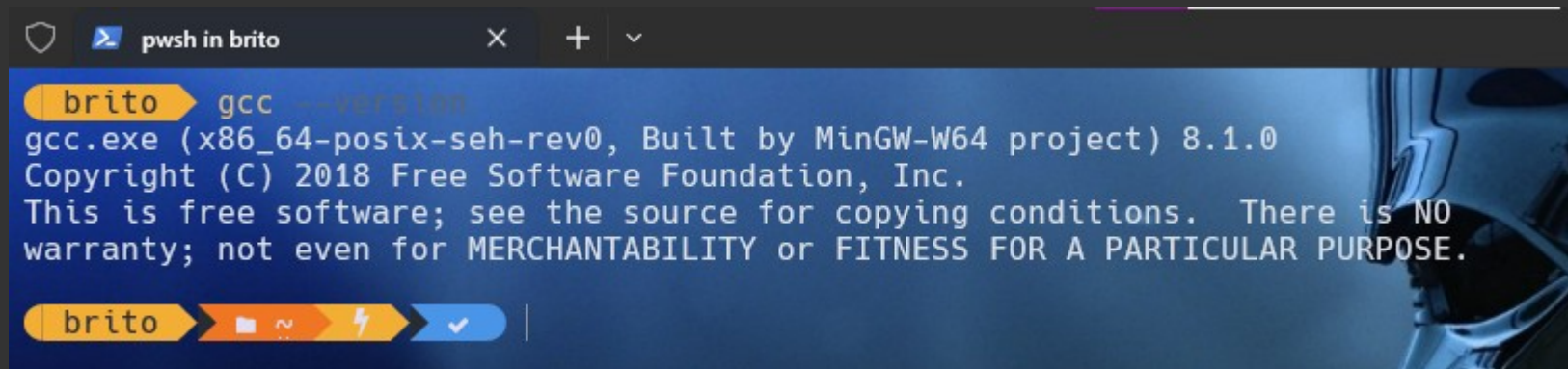
Ao compilar a partir do repositório de origem ou instantâneos, ou se você modificar a documentação do Texinfo, precisará da versão 4.7 ou posterior do Texinfo instalada se desejar que a documentação do Info seja regenerada. Os lançamentos contêm documentação de informações pré-criada para a documentação não modificada no lançamento.

Construindo um compilador nativo

Para uma compilação nativa, a configuração padrão é executar uma inicialização de 3 estágios do compilador quando `'fazer'` é invocado. Isso criará todo o sistema GCC e garantirá que ele seja compilado corretamente. Pode ser desabilitado com o `--disable-bootstrap` parâmetro para `'configurar'`, mas a inicialização é sugerida porque o compilador será testado de forma mais completa e também poderá ter



Versão do gcc



```
pwsh in brito  
brito> gcc --version  
gcc.exe (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0  
Copyright (C) 2018 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
brito>
```

Alem de instalado ele deve ser
Adicionado ao path

```
pwsh in GNU-AULAS
```

-a----	11/12/2022	15:08	19983956	kupdf.net_linguagem-c-completa-e-descomplicada.pdf
-a----	30/08/2022	23:03	138	leramanha.txt
-a----	30/08/2022	00:46	1756	libMinha.txt
-a----	02/10/2022	22:20	1553459	Livro-de-Enoque-1-e-2.pdf
-a----	17/09/2022	16:26	41000	MILENIOS-DE-GUERRAS.pdf
-a----	30/09/2022	03:31	36	musasbrasil.txt
-a----	02/08/2022	14:01	4053649	node-express-course-main.zip
-a----	02/10/2022	22:17	458278	0-Livro-de-Enoque.pdf
-a----	19/09/2022	09:21	309760	OpenGL.doc
-a----	29/08/2022	00:35	158053	PROG1_16_Registros_em_C.pdf
-a----	30/08/2022	15:07	720703	Programacao Aplicada em C pra Windows.pdf
-a----	24/12/2022	16:38	231	PUREFER.txt
-a----	26/08/2022	02:26	941411	raylib_cheatsheet_v4.2.pdf
-a----	02/08/2022	18:38	311	RICKANDMORTY.txt
-a----	23/07/2021	17:46	10394	scarfy.png
-a----	07/07/2022	11:20	21261	scarfy_run.gif
-a----	12/09/2022	16:18	489	tutorclipstudio.txt
-a----	01/09/2022	22:10	1437033	Tutorial_OpenGL.PDF
-a----	24/08/2022	17:06	15399705	UM03001_emWin5.pdf
-a----	01/09/2022	23:21	253	v8.txt

```
F:\> md GNU-AULAS
```

Diretório: F:\

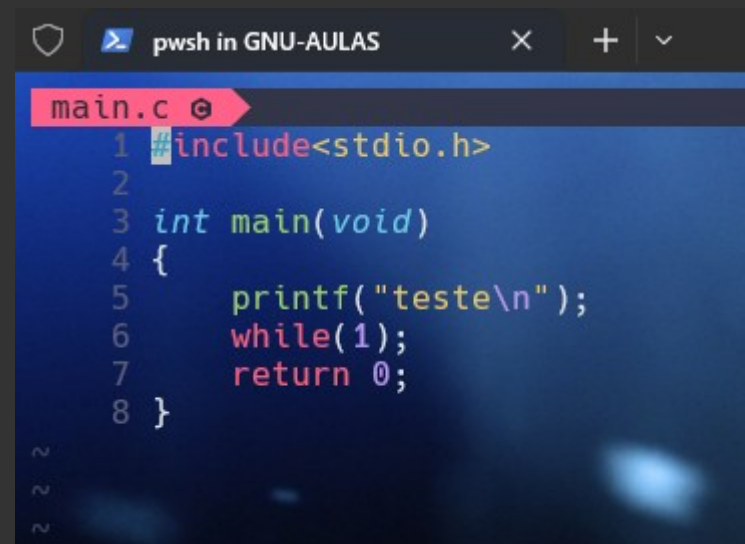
Mode	LastWriteTime	Length	Name
d----	20/02/2023 13:16		GNU-AULAS

```
F:\> CD GNU-AULAS
GNU-AULAS> ls
brito> GNU-AULAS> nvim main.c
```

in pwsh at 13:16:16

Criei um diretorio chamado GNU-AULAS no drive f:
Dentro dele o arquivo main.c
E editei ele com o nvim

No nvim
Esc para sair do modo e :wq para salvar e sair



The screenshot shows a terminal window with a dark theme. The title bar at the top reads "pwsh in GNU-AULAS". Below the title bar, a file named "main.c" is open in a text editor. The code in the editor is as follows:

```
1 #include<stdio.h>
2
3 int main(void)
4 {
5     printf("teste\n");
6     while(1);
7     return 0;
8 }
```

On the left side of the editor, there are three tilde (~) symbols, likely representing a scrollbar or line indicators.

Arquivo main.c

pwsh in GNU-AULAS

-a----	26/08/2022	02:26	941411	raylib_cheatsheet_v4.2.pdf
-a----	02/08/2022	18:38	311	RICKANDMORTY.txt
-a----	23/07/2021	17:46	10394	scarfy.png
-a----	07/07/2022	11:20	21261	scarfy_run.gif
-a----	12/09/2022	16:18	489	tutorclipstudio.txt
-a----	01/09/2022	22:10	1437033	Tutorial_OpenGL.PDF
-a----	24/08/2022	17:06	15399705	UM03001_emWin5.pdf
-a----	01/09/2022	23:21	253	v8.txt

F:\ md GNU-AULAS

Diretório: F:\

Mode	LastWriteTime	Length	Name
d----	20/02/2023 13:16		GNU-AULAS

F:\ CD GNU-AULAS

GNU-AULAS ls

GNU-AULAS nvim main.c

GNU-AULAS ls

Diretório: F:\GNU-AULAS

Mode	LastWriteTime	Length	Name
-a----	20/02/2023 13:21	97	c main.c

brito GNU-AULAS

in pwsh at 13:21:48

31°C Parc ens...

Temos o arquivo main.c

```
Diretório: F:\

Mode                LastWriteTime         Length Name
----                -
d-----          20/02/2023   13:16             GNU-AULAS

F:\> CD GNU-AULAS
GNU-AULAS> ls
GNU-AULAS> nvim main.c
GNU-AULAS> ls

Diretório: F:\GNU-AULAS

Mode                LastWriteTime         Length Name
----                -
-a-----          20/02/2023   13:21           97 c  main.c

brito GNU-AULAS gcc -o executavel main.c
```

-o executavel é o nome que queremos dar
Main.c é o nosso arquivo
Na pagina diz que a ordem não importa -o


```
F:\> CD GNU-AULAS
GNU-AULAS> ls
GNU-AULAS> nvim main.c
GNU-AULAS> ls

Diretório: F:\GNU-AULAS

Mode                LastWriteTime         Length Name
----                -
-a----             20/02/2023   13:21           97 c    main.c

GNU-AULAS> gcc -o executavel main.c
GNU-AULAS> ls

Diretório: F:\GNU-AULAS

Mode                LastWriteTime         Length Name
----                -
-a----             20/02/2023   13:27   54022 ❏ executavel.exe
-a----             20/02/2023   13:21           97 c    main.c

brito > GNU-AULAS
```

Com ls vemos que o arquivo foi compilado

```
Diretório: F:\GNU-AULAS

Mode                LastWriteTime         Length Name
----                -
-a----            20/02/2023   13:21             97 c   main.c

GNU-AULAS gcc -o executavel main.c
GNU-AULAS ls

Diretório: F:\GNU-AULAS

Mode                LastWriteTime         Length Name
----                -
-a----            20/02/2023   13:27        54022 □ executavel.exe
-a----            20/02/2023   13:21             97 c   main.c

GNU-AULAS ./executavel.exe
teste
█
```

Executamos sem problemas
Precisamos estar na pasta do projeto
./executavel.exe

GNU-AULAS gcc --help

Usage: gcc.exe [options] file...

Options:

-pass-exit-codes	Exit with highest error code from a phase.
--help	Display this information.
--target-help	Display target specific command line options.
--help={common optimizers params target warnings [^]{joined separate undocumented}}[,...].	Display specific types of command line options.
(Use '-v --help' to display command line options of sub-processes).	
--version	Display compiler version information.
-dumpspecs	Display all of the built in spec strings.
-dumpversion	Display the version of the compiler.
-dumpmachine	Display the compiler's target processor.
-print-search-dirs	Display the directories in the compiler's search path.
-print-libgcc-file-name	Display the name of the compiler's companion library.
-print-file-name=<lib>	Display the full path to library <lib>.
-print-prog-name=<prog>	Display the full path to compiler component <prog>.
-print-multiarch	Display the target's normalized GNU triplet, used as a component in the library path.
-print-multi-directory	Display the root directory for versions of libgcc.
-print-multi-lib	Display the mapping between command line options and multiple library search directories.
-print-multi-os-directory	Display the relative path to OS libraries.
-print-sysroot	Display the target libraries directory.
-print-sysroot-headers-suffix	Display the sysroot suffix used to find headers.
-Wa,<options>	Pass comma-separated <options> on to the assembler.
-Wp,<options>	Pass comma-separated <options> on to the preprocessor.
-Wl,<options>	Pass comma-separated <options> on to the linker.
-Xassembler <arg>	Pass <arg> on to the assembler.
-Xpreprocessor <arg>	Pass <arg> on to the preprocessor.
-Xlinker <arg>	Pass <arg> on to the linker.
-save-temps	Do not delete intermediate files.
-save-temps=<arg>	Do not delete intermediate files.
-no-canonical-prefixes	Do not canonicalize paths when building relative prefixes to other gcc components.
-pipe	Use pipes rather than intermediate files.

Gcc -help
Vamos o menu de ajuda

CADA PESSOA TEM UM CAMINHO QUE SÓ ELA SABE COMO TRILHAR
PENSANDO NISSO

É

MELHOR SABER LER A API E A DOC

De qualquer linguagem que se queira aprender

Pois temos desde as atualizações até os bugs



