

C言語基礎演習('20)／続・C言語基礎演習('23)

学習参考情報

ここでは、放送大学オンライン授業「C言語基礎演習('20)」と「続・C言語基礎演習('23)」の学習参考情報のページです。よくある(と思われる)質問と回答や、授業に関連した情報を掲載します。

よくある(と思われる)質問と回答

C言語の環境構築

この科目は、C言語の開発環境をご自身で用意できることを履修の条件としております。[オンライン授業体験版](#)の「C言語基礎演習('20)」の体験コースで、環境構築の方法を説明しています。体験版やこのページを参考に環境構築、動作確認を行ってから履修してください。

gccを実行しても「fatal error」と表示され正しく実行できません。

gccの書式、ソースファイルの名前や保存場所などが誤っている場合は、「fatal error」と表示されます。これはgccからのエラーメッセージで、gcc自体が実行できていないわけではありません。

実行ファイルを実行しようとしても「アクセスが拒否されました。」等と表示され、実行できません。

セキュリティソフトウェア(ウイルス対策ソフト)によっては、コンパイルして作った実行ファイルをウイルスと判定(誤判定)し、実行をブロックしたり、実行ファイルを削除したりすることがあるようです。演習で使うフォルダ(「C言語基礎演習('20)」での推奨通りに作成した場合は、C:\%c_ensyu)を、スキャン対象から除外する設定がないかご確認ください。

実行ファイルを実行しても何も起こりません。または、実行してもすぐに終了します。

実行ファイルをダブルクリックして実行していませんか。「C言語基礎演習('20)」の[TeraPad・MinGW-w64のインストール方法と使い方 » 2.3 コマンドプロンプトの操作とコンパイル] に記載したように、コマンドプロンプトに実行ファイル名を入力することによって実行します。

printf() 関数で ¥n を表示すると、改行されるのではなく、「¥n」がそのまま表示されます。

円記号またはバックスラッシュに見える文字「¥」と、円記号に見える文字「¥」があります。改行のために使う「¥」は前者の文字で、Macのキーボードで入力される「¥」は後者の文字です。option + ¥ で前者の文字が入力できます。

MinGW-w64以外の開発環境

Windowsで使えるMinGW-w64以外の開発環境として、Windows Subsystem for Linux (WSL) を簡単に紹介します。

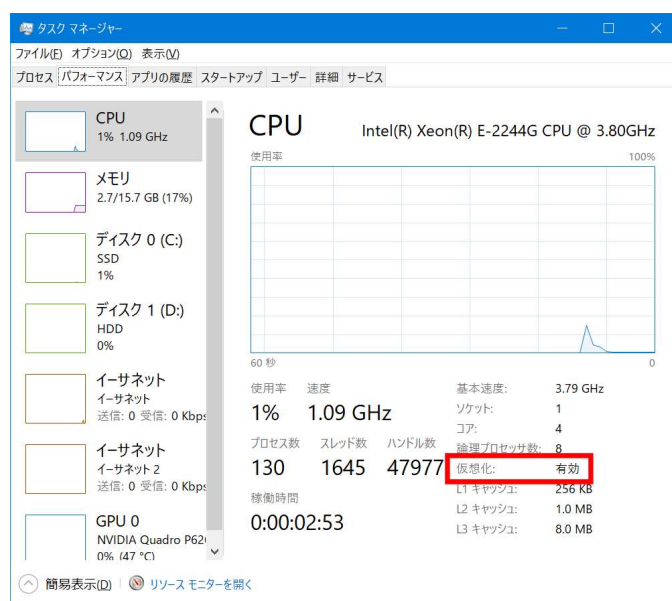
Windows Subsystem for Linux (WSL)

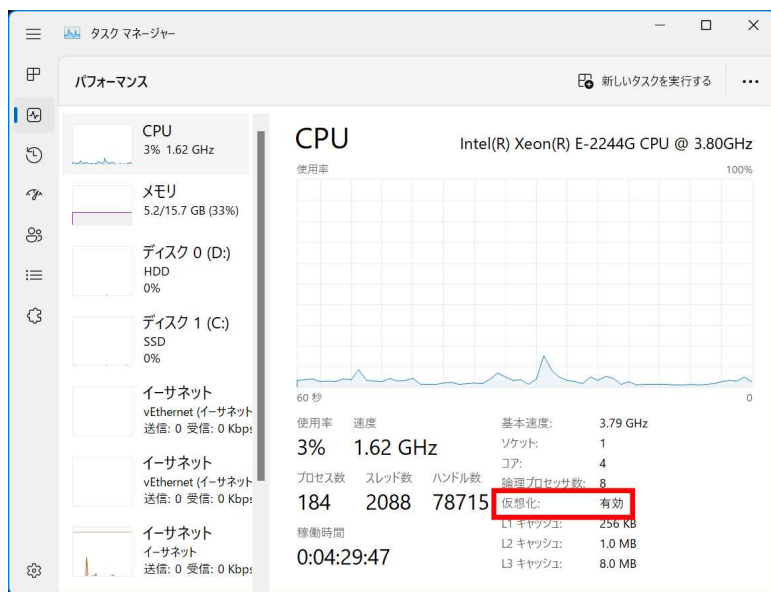
以下は、2022年9月時点でのWSL 2の情報です。

インストール前の確認

タスクマネージャーを起動します。左下に「詳細(D)」と表示されている場合は、それをクリックしてください。「簡易表示(D)」と表示されている場合や、いずれも見当たらない場合は、何もする必要はありません。

「パフォーマンス」タブ、もしくはサイドバーの「パフォーマンス」を選択します。左のペインで「CPU」が選択されていることを確認し、右のペインで「仮想化」が「有効」になっていることを確認します。





Ubuntuのインストール

コマンドプロンプトを管理者として実行します。

「`wsl --install -d Ubuntu`」と入力します。「変更を有効にするには、システムを再起動する必要があります。」と表示されたら、Windowsを再起動します。

```
管理者: コマンド プロンプト
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>wsl --install -d Ubuntu
インストール中: 仮想マシン フラットコンテナ
仮想マシン プラットフォーム はインストールされました。
インストール中: Linux 用 Windows サブシステム
Linux 用 Windows サブシステム はインストールされました。
ダウンロード中: Ubuntu
要求された操作は正常に終了しました。変更を有効にするには、システムを再起動する必要があります。

C:\Windows\system32>
```

再起動後、インストールが自動的に再開します。しばらく時間がかかります。

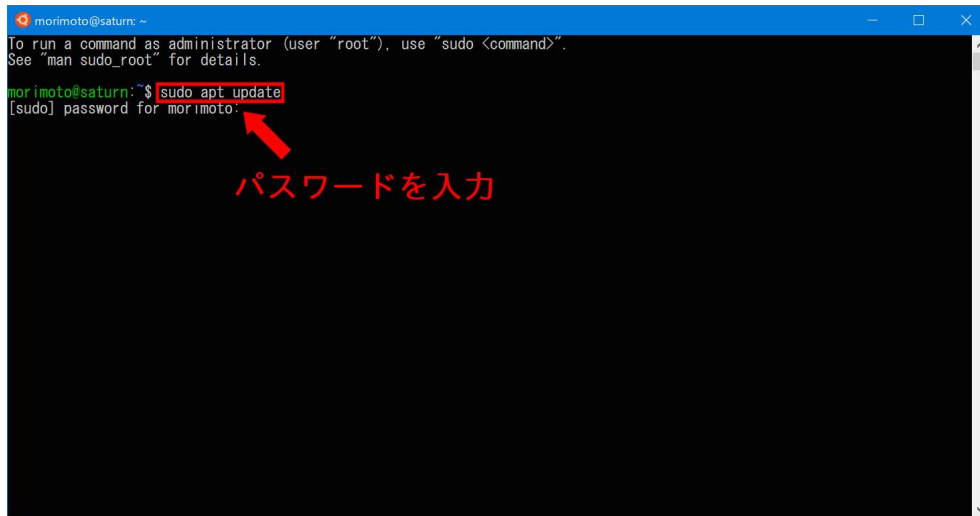
新しいユーザのユーザ名とパスワードを考え、「Enter new UNIX username:」にユーザ名を、「New password:」と「Retype new password:」にパスワードを入力します。ここで入力するユーザ名とパスワードは、Windowsのものとは無関係です。

Ubuntuを使用するには、スタートメニューから「Ubuntu」(または「Ubuntu on Windows」)のような名称のアプリ)を選択します。使用を終えるには、ウィンドウ(ターミナル)に「`exit`」と入力します。

インストール後の環境構築

以降、sudoから始まるコマンドを入力した際にパスワードを聞かれたら、インストール時に設定したパスワードを入力します。

「sudo apt update」と入力します。

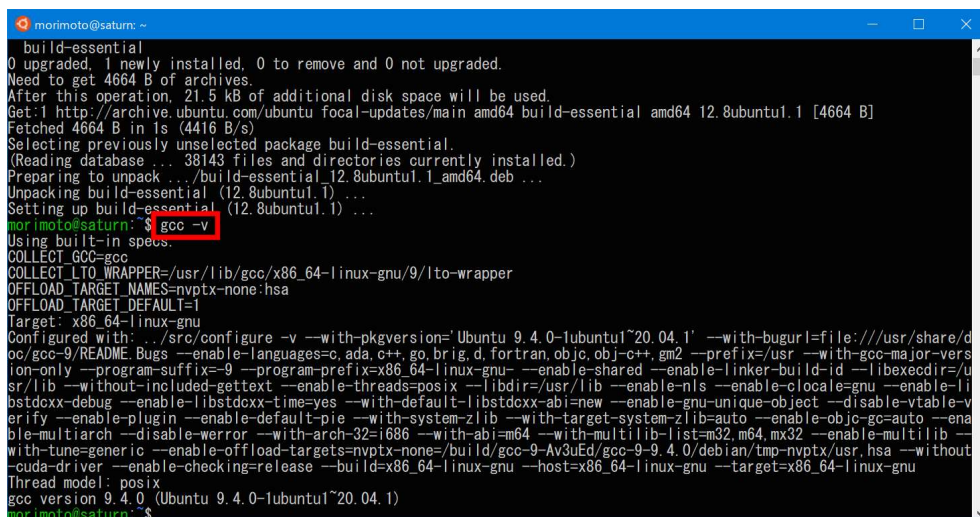


```
morimoto@saturn: ~  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
morimoto@saturn:~$ sudo apt update  
[sudo] password for morimoto:
```

「apt list --upgradable」と入力します。「Listing... Done」しか表示されなければ、何もする必要はありません。そうでなければ、「sudo apt upgrade」と入力し、「Do you want to continue? [Y/n]」でエンターキーを押します。

「sudo apt install build-essential」と入力し、「Do you want to continue? [Y/n]」でエンターキーを押します。

「gcc -v」と入力し、コンパイラがインストールされたことを確認します。



```
morimoto@saturn: ~  
build-essential  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 4664 B of archives.  
After this operation, 21.5 kB of additional disk space will be used.  
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 build-essential amd64 12.8ubuntu1.1 [4664 B]  
Fetched 4664 B in 1s (4416 B/s)  
Selecting previously unselected package build-essential.  
(Reading database ... 38143 files and directories currently installed.)  
Preparing to unpack .../build-essential_12.8ubuntu1.1_amd64.deb ...  
Unpacking build-essential (12.8ubuntu1.1) ...  
Setting up build-essential (12.8ubuntu1.1) ...  
morimoto@saturn:~$ gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper  
OFFLOAD_TARGET_NAMES=rvv-none:hsa  
OFFLOAD_TARGET_DEFAULT=1  
Target: x86_64-linux-gnu  
Configured with: ./src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04.1' --with-bugurl=file:///usr/share/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-9 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale-gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch=32-i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=rvv-none:/build/gcc-9-Av3uEd/gcc-9-9.4.0/debian/tmp-nvptx/usr,hsa --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu  
Thread model: posix  
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.1)  
morimoto@saturn:~$
```

この授業での使い方

cd コマンドでカレントディレクトリを移動します。Windowsのフォルダをそのまま指定することはできません。「wslpath "Windowsのフォルダのパス"」で表示されるディレクトリを指定します。下の画像では、「wslpath "C:\c_ensyu"」と入力し、「/mnt/c/c_ensyu」と表示されています。この場合、「cd /mnt/c/c_ensyu」でカレントディレクトリを移動します。

```
morimoto@saturn: /mnt/c/c_ensyu
Setting up libstdc++-9-dev:amd64 (9.4.0-1ubuntu1~20.04.1) ...
Setting up gcc (4:9.3.0-1ubuntu2) ...
Setting up g++-9 (9.4.0-1ubuntu1~20.04.1) ...
Setting up g++ (4:9.3.0-1ubuntu2) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.8ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
morimoto@saturn:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04.1' --with-bugurl=file:///usr/share/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-9 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch=32-i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none=/build/gcc-9-Av3uEd/gcc-9-9.4.0/debian/tmp-nvptx/usr,hsa --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
morimoto@saturn:~$ wslpath "C:\c_ensyu"
/mnt/c/c_ensyu
morimoto@saturn:~$ cd /mnt/c/c_ensyu
morimoto@saturn:/mnt/c/c_ensyu$
```

ex01.c をコンパイルするには、「gcc -o ex01 ex01.c」と入力します。実行ファイル名に拡張子「.exe」を付けません(付けても構いませんが一般的ではありません)。実行ファイルを実行するには、「./ex01」と入力します。実行ファイル名(ex01)の先頭に「./」(ドット・スラッシュ)を付けます。

```
morimoto@saturn: /mnt/c/c_ensyu
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
morimoto@saturn:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04.1' --with-bugurl=file:///usr/share/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-9 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch=32-i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none=/build/gcc-9-Av3uEd/gcc-9-9.4.0/debian/tmp-nvptx/usr,hsa --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
morimoto@saturn:~$ wslpath "C:\c_ensyu"
/mnt/c/c_ensyu
morimoto@saturn:~$ cd /mnt/c/c_ensyu
morimoto@saturn:/mnt/c/c_ensyu$ gcc -o ex01 ex01.c
morimoto@saturn:/mnt/c/c_ensyu$ ./ex01
こんにちは
morimoto@saturn:/mnt/c/c_ensyu$ gcc -o ex02 ex02.c
morimoto@saturn:/mnt/c/c_ensyu$ ./ex02
今何歳ですか? > 42
6年後には47歳です。
morimoto@saturn:/mnt/c/c_ensyu$
```

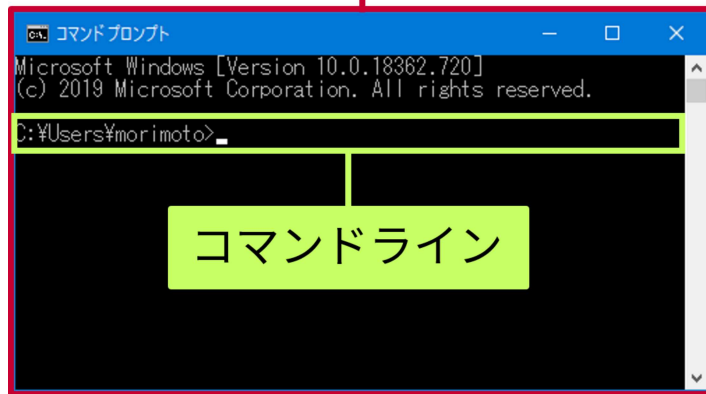
ソースファイルの文字コードはUTF-8にしてください。授業のサイトの「ソースコード一式 (UTF-8, LF)」を使ってください。

その他の情報

コマンドプロンプトの操作

ここでは、コマンドプロンプトの中で、コマンドを入力する行をコマンドラインと呼びます。コマンドラインの前半はプロンプトです。

コマンドプロンプト



C言語のプログラムは、一度の記述で正しくコンパイル、実行できることは多くありません。エラーメッセージや動作を確認し、ソースコードを修正しながら何度もコンパイル、実行する必要があります。その際、「gcc -o ex01.exe ex01.c」などと何度も入力するのは非効率です。これまでに入力したコマンドを再度呼び出すなどして、作業を省力化できる便利な機能を紹介します。

コマンドプロンプトで`↑`キーを押すと、直前に呼び出したコマンドがコマンドラインに入力された状態になります。その状態でエンターキーを押すと、そのコマンドが実行できます。`↑`キーを複数回押すと、1回前、2回前、…のように、コマンドの履歴をたどることができます。`↓`キーは、コマンドの履歴を`↑`キーとは逆にたどります。

`↑`キーと`↓`キーを使って以前に入力したコマンドを表示させた後、`←`キーと`→`キーでカーソルの位置を移動して、編集することができます。たとえば、「gcc -o ex01.exe ex01.c」の「1」を2箇所消して「2」にすれば、「gcc -o ex02.exe ex02.c」と入力した状態にできます。

`F7`キーを押すと、これまでに実行したコマンドの履歴が表示されます。`↑`キーと`↓`キーで選択し、エンターキーを押すと、そのコマンドが実行されます。履歴の選択後、エンターキーではなく`→`キーを押すと、そのコマンドがコマンドラインに入力された状態になります。

コマンドラインに入力した文字列をすべて消したい場合は、`ESC`キーを押してください。