Setup PhysiCell on Linux

Elmar Bucher, Ph.D.-Student

Intelligent Systems Engineering
Indiana University

2025-01-25

Overview

This document describes the PhysiCell installation on a Debian Linux distribution.
If you run on another flavor, please adjust accordingly.

- GCC
- ImageMagick, FFmpeg, Tar, Gzip, Unzip
- Git
- PhysiCell

} Basic Setup

- Python3, iPython, pcdl
- PhysiCell Studio

} Extended Setup

- VSCode (if you are not a Vim or Emacs user)

} IDE Setup

Apt (Basic Setup)

- First, let's update and upgrade apt, the package manager.

```
sudo apt update
sudo apt upgrade
```

GCC (Basic Setup)

- Check if GCC (the gnu compiler collection) is already installed.

```
g++ --version
```

- If not:

```
sudo apt install build-essential
```

- OpenMP (open multi processing) is a feature, used by PhysiCell, that is implemented in GCC.

ImageMagick part I (Basic Setup)

ImageMagick is used for making jpeg and gif images from PhysiCell svg image output.

- Check if ImageMagick is already installed.

  ```
  magick --version
  ```

  - **If** ok: **you have >= version 7 installed**. You are all set! You can move to the next page.
  - **If** you receive: `Command 'magick' not found`, try:

    ```
    convert --version
    ```

    If ok: **you have most probably a version 6 installed** (like the most of us)!
  - **If** both of these commands do not work:

    ```
    sudo apt install imagemagick
    ```

ImageMagick part II (Basic Setup)

The PhysiCell `Makefile` is written for ImageMagick >= version 7, which requires a `magick` command in front of each ImageMagick command (e.g. `magick convert` instead of `convert`).

If and only if you have <= version 6 installed, you can follow the instruction below to generate a `magick` command that simply passes everything to the next command. This will make the PhysiCell `Makefile` work for you too.

`cd` to the folder where you have your manual installed binaries. e.g. `~/.local/bin/`

```
echo '$*' > magick
chmod 775 magick
which magick
```

FFmpeg (Basic Setup)

FFmpeg is used by the `Makefile` and pcdl for making mp4 movies out of jpeg images.

- Check if FFmpeg is already installed.
  ```
  ffmpeg -version
  ```

- If not:
  ```
  sudo apt install ffmpeg
  ```

Tar Gzip Unzip (Basic Setup)

Tar, Gzip, and Unizip are used by the `Makefile` to zip, tar, unzip, and untar folders.

- Check if you have `tar, gzip, and unzip` installed.

```
tar --version
gzip --version
unzip --version
```

- if not:

```
sudo apt install tar gzip unzip
```

Git (Basic Setup)

`git` is the power user way to install the PhysiCell and the Studio source code.

- Check if git is already installed
  ```
  git --version
  ```

- If not:
  ```
  sudo apt install git
  ```

PhysiCell (Basic Setup)

Download PhysiCell and place it e.g in the ~/src folder.

```
mkdir -p ~/src
cd ~/src
git clone https://github.com/MathCancer/PhysiCell.git
```

Test the installation with the template sample project.

```
cd PhysiCell
make template
make -j8
./project
make jpeg
make gif
make movie
```

Overview

This document describes the PhysiCell installation on a Debian Linux distribution.
If you run on another flavor, please adjust accordingly.

- GCC
- ImageMagick, FFmpeg, Tar, Gzip, Unzip
- Git
- PhysiCell

Basic Setup

- Python3, iPython, pcdl
- PhysiCell Studio

Extended Setup

- VSCode (if you are not a Vim or Emacs user)

IDE Setup

Python3 part I (Extended Setup)

We will **generate a python3 environment with the default python installation**, where we will install all PhysiCell modelling related python libraries. We will name this python3 environment **pcvenv** (PhysiCell virtual environment), and we install it in the src folder where just before have installed PhysiCell.
Here we demonstrate, how to generate the environment with the regular python. If you run mamba or conda, please adjust the commands accordingly.

- Check if python3 and pip are installed:

```
which python3
which pip3
```

- Python will be installed, but pip might be missing!

```
sudo apt install python3-pip
```

Python3 part II (Extended Setup)

- Generate a python environment named pcvenv (PhysiCell Python environment):

```
cd ~
python3 -m venv src/pcvenv
```

- Generate an alias for this environment for activation:

```
echo "alias pcvenv=\"source /home/$USER/src/pcvenv/bin/activate\"" >> ~/.bash_aliases
source ~/.bash_aliases
```

Python3 part III (Extended Setup)

- Activate the pcvenv python environment using the alias generated before:

```
pcvenv
```

- Check if the python and pip paths point to the installed location, the python and pip in the environment:

```
which python3
which pip3
```

- Install the iPython shell:

```
pip3 install ipython
```

- Install the PhysiCell Data Loader:

```
pip3 install pcdl
```

PhysiCell Studio part I (Extended Setup)

- Install the studio Qt library dependencies:

```
sudo apt install qtbase5-dev
```

- Download the studio, place it in the src folder, too, and install its python3 dependencies.

```
cd ~/src
git https://github.com/PhysiCell-Tools/PhysiCell-Studio.git
pip3 install -r PhysiCell-Studio/requirements.txt
```

- Put the studio under the `PATH`:

`cd` to the folder where you have your manual installed binaries. e.g. `~/.local/bin/`

```
echo "python3 /home/$USER/src/PhysiCell-Studio/bin/studio.py \$*" > pcstudio
chmod 775 pcstudio
which pcstudio
cd ~
```

PhysiCell Studio part II (Extended Setup)

- Test the installation with the template sample project.

```
cd ~/src/PhysiCell
pcvenv
pcstudio
```

  PhysiCell Studio should open and load the template PhysiCell_settings.xml file.

- Please check out the official PhysiCell Studio manual:
  https://github.com/PhysiCell-Tools/Studio-Guide/tree/main

Overview

This document describes the PhysiCell installation on a Debian Linux distribution.
If you run on another flavor, please adjust accordingly.

- GCC
- ImageMagick, FFmpeg, Tar, Gzip, Unzip
- Git
- PhysiCell

Basic Setup

- Python3, iPython, pcdl
- PhysiCell Studio

Extended Setup

- VSCode (if you are not a Vim or Emacs user)

IDE Setup

MS Visual Studio Code part I (IDE Setup)

1. Install vs code, either from your operating system's app store or from https://code.visualstudio.com/

2. Generate a vs code profile for physicell:

```
File | New Window with Profile
Name: physicell
Icon: choose a cool one. e.g. 🔥.
Create
Add Folder: Home/src
click the profile icon (default is a gearwheel) on the left side bottom corner.
Profile > physicell
```

3. Open the Folder:

```
File | Open Folder… | src | Open
Yes, I trust the authors
```

🐧 🐧 🐧

MS Visual Studio Code part II (IDE Setup)

1. Install the official python and C++ extensions into the profile:

   ```
   click the profile icon (default is a gearwheel) on the left side bottom corner.
   Profile > physicell
   Extension: Python Install
   Extension: C/C++ Install
   ```

2. Link pcvenv (the python environment we generated above):

   ```
   View | Command Palette… | Python: Select Interpreter |
   Enter interpreter path… | Find… | src/pcvenv
   ```

Overview

This document describes the PhysiCell installation on a Debian Linux distribution.
If you run on another flavor, please adjust accordingly.

- GCC
- ImageMagick, FFmpeg, Tar, Gzip, Unzip
- Git
- PhysiCell

Basic Setup

- Python3, iPython, pcdl
- PhysiCell Studio

Extended Setup

- VSCode (if you are not a Vim or Emacs user)

IDE Setup

Miniforge Python3 part I (Extended Setup)

The **mamba** package manager (faster and robuster than conda, otherwise the same and totally compatible with pip and conda) comes in handy, when you want to run somewhere python3 where you have no root rights, or if you prefer to leave the python3 that ships with the distribution untouched.
https://mamba.readthedocs.io/en/latest/user_guide/mamba.html

- `cd` to the folder where you have your manual installed programs.
  e.g. `~/.local/lib/`

- Adjust the download according to your CPU architecture (here x86_64): https://github.com/conda-forge/miniforge

- `wget https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-Linux-x86_64.sh`
  `chmod 775 Miniforge3-Linux-x86_64.sh`
  `./Miniforge3-Linux-x86_64.sh`

- Adjust the installation location to where your manual installed programs are.
  e.g. `home/$USER/.local/lib/miniforge3`

- You wish the installer to initialize mamba!
  `yes`
  `source ~/.bashrc` (or log out and in again).

Miniforge Python3 part II (Extended Setup)

- With **mamba** you can now generate virtual python environment, in addition to the (base) environment, with the command below. **Beware**: You have at least to list one package (e.g. ipython) to successfully generate an environment!

```
mamba create -n myenv ipython
```

- You can list all existing virtual environments like this:

```
mamba env list
```

- You can activate a virtual environment like this:

```
mamba activate myenv
```

- To escape from all virtual environments, e.g. to work with the python that ships with the operating system, like this:

```
mamba deactivate
```

Acknowledgement

The first version of this installation manual was written for the summer workshop in 2022 by:

★ Aneequa Sundus (Windows)
★ Furkan Kurtoglu (Windows)
★ John Metzcar (Apple)
★ Randy Heiland (Apple)