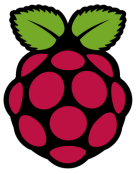
CHARLES CARNUS
LYCÉE TECHNIQUE
ENSEIGNEMENT SUPÉRIEUR



TP Raspberry Pi

C, C++, Python ...

Prof : **Kamal Boudjelaba**

25 août 2025

B T S S N

LTP Charles Carnus
Avenue de Saint Pierre - Bourran - 12000 RODEZ, France

IR–EC

Table des matières

1. Introduction : commandes Linux Raspberry Pi

Note : Le symbole $ indique une commande à exécuter dans le terminal. Certaines commandes nécessitent les droits administrateur et doivent être précédées de `sudo`. est utilisé pour présenter une commande, et pour donner un exemple d'exécution.

Commande	Signification
`$ raspi-config`	Outil de configuration pour Raspberry Pi
`$ raspistill`	Prendre une photo avec le module caméra du Raspberry Pi
`$ apt`	Gestionnaire de paquets sur Raspberry Pi OS
`$ ifconfig`	Affiche la configuration réseau actuelle
`$ nano`	Éditeur de fichiers texte (en mode terminal) par défaut sur Raspberry Pi OS
`$ wget`	Téléchargement de fichiers depuis Internet

Commande	Signification & Exemples
`$ apt-get update`	Met à jour les informations des dépôts de paquets `$ sudo apt-get update`
`$ apt-get upgrade`	Met à jour les paquets installés `$ sudo apt-get upgrade`

`$ apt-get install <paquet>`	Installe un ou plusieurs paquets spécifiés
	
	`$ sudo apt-get install conda`
	
	`$ sudo apt-get install conda pip`

`$ apt-get remove <paquet>`	Désinstalle un paquet `$ sudo apt-get remove conda`

`$ dpkg -l`	Liste tous les paquets installés. Utiliser `grep` pour filtrer
	
	`$ dpkg -l`
	
	`$ dpkg -l | grep ...`

Commande	Signification & Exemples
`$ cd <chemin>`	Change le répertoire courant (se déplace vers le dossier indiqué) `$ cd /home/pi`

`$ ls`	Liste les fichiers et dossiers du répertoire actuel (ou spécifié)
	
	`$ ls`
	
	`$ ls /home/pi`

`$ mkdir <dossier>`	Crée un dossier dans l'emplacement actuel ou spécifié
	
	`$ mkdir MonDossier`
	
	`$ mkdir /home/pi/MonDossier`

`$ cp <source> <destination>`	Copie un fichier ou un dossier vers un autre emplacement
	
	`$ cp test.txt /home/pi/Documents/`
	
	`$ cp /home/pi/test.txt /home/pi/Documents/`

`$ mv <source> <destination>`	Déplace ou renomme un fichier ou un dossier
	
	`$ mv /home/pi/test.txt /home/Documents/`
	
	`$ mv /home/pi/MonDossier/ /home/Documents/`

`$ cat <fichier>`	Affiche le contenu d'un fichier dans le terminal `$ cat /home/pi/test.txt`

`$ rm <fichier>`	Supprime un fichier. Pour supprimer un dossier, ajouter `-rf`
	
	`$ rm test.txt`
	
	`$ rm -rf /home/pi/MonDossier/`

`$ pwd`	Affiche le chemin du répertoire courant `$ pwd`
`$ tree`	Affiche l'arborescence des dossiers à partir du répertoire actuel `$ tree`

Commande	Signification & Exemples
`$ tar -c`	Utilise la commande `tar` pour créer une archive (souvent compressée avec gzip) `$ tar -cvfz archive.tar.gz /home/pi/Documents/MonDossier` `-c` : création d'une archive `-v` : mode verbeux `-f` : spécifie le nom de l'archive `-z` : compression avec gzip
`$ tar -x`	Extrait les fichiers d'une archive tar.gz `$ tar -xvfz archive.tar.gz`

2. Écrire et exécuter des programmes sur le Raspberry Pi (Linux ARM)

2.1 Introduction : Éditeur de texte nano

Nano est un éditeur de texte simple mais puissant, conçu pour les systèmes Unix et Linux. Il permet de créer ou modifier facilement des fichiers texte.

Certains systèmes (comme macOS et Linux) incluent généralement Nano par défaut. Pour vérifier s'il est installé :

```
nano --version
```

Pour l'installer sur Debian ou Ubuntu :

```
sudo apt-get install nano
```

— Démarrer Nano sans ouvrir de fichier spécifique :

```
sudo nano
```

— Créer ou modifier un fichier existant :

```
sudo nano Nom_Fichier.Extension
```

Exemple :

```
sudo nano prog1.c
```

— Si le fichier n'est pas dans le répertoire courant, préciser le chemin complet :

```
sudo nano /home/pi/MesProgrammes/prog2.py
```

Si le fichier spécifié n'existe pas, Nano le créera automatiquement. Si aucun nom n'est indiqué, un fichier temporaire sera créé, et Nano demandera un nom lors de la fermeture.

Au bas de la fenêtre, Nano affiche des raccourcis clavier. Le symbole ∧ indique une combinaison avec la touche CTRL (ex. : ∧O = CTRL + O).

Pour enregistrer : CTRL + O Pour quitter : CTRL + X Si des modifications ont été faites, Nano demandera si vous souhaitez les enregistrer : appuyez sur Y (oui), ou N (non), puis Entrée.

Raccourci	Fonction
CTRL + A	Aller au début de la ligne
CTRL + E	Aller à la fin de la ligne
CTRL + Y	Faire défiler vers le **haut**
CTRL + V	Faire défiler vers le **bas**
CTRL + G	Ouvrir l'aide de Nano
CTRL + O	Sauvegarder le fichier. Permet de modifier ou confirmer le nom avant enregistrement
CTRL + W	Rechercher une expression. Pour relancer la recherche, utiliser ALT + W
CTRL + K	Couper la ligne courante
CTRL + U	Coller une ligne précédemment coupée
CTRL + X	Quitter Nano (demande de sauvegarde si nécessaire)

2.2 Programme en C

1. Créer un dossier nommé `ProgNano` dans `/home/pi`
2. Créer un fichier nommé `ProgC.c` :

```
sudo nano /home/pi/ProgNano/ProgC.c
```

3. Écrire le programme suivant, enregistrer puis quitter Nano :

```c
#include <stdio.h>

int main(void)
{
    printf("Le programme en C nommé ProgC.c et écrit dans nano a été exécuté.\n");
    return 0;
}
```

4. Compiler le programme :

```
gcc –o TestC ProgC.c
```

5. Exécuter le programme :

```
./TestC
```

2.3 Programme en C++

1. Créer un fichier nommé `ProgCPP.cpp` dans `/home/pi/ProgNano` :

```
sudo nano /home/pi/ProgNano/ProgCPP.cpp
```

2. Écrire le programme suivant, enregistrer puis quitter Nano :

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    cout << "Le programme en C++ nommé ProgCPP.cpp et écrit dans nano a été exécuté." << endl;
    return 0;
}
```

3. Compiler le programme :

```
g++ –o TestCPP ProgCPP.cpp
```

4. Exécuter le programme :

```
./TestCPP
```

2.4 Programme en Python

1. Créer un fichier nommé `ProgPython.py` dans `/home/pi/ProgNano` :

```
sudo nano /home/pi/ProgNano/ProgPython.py
```

2. Écrire le programme suivant, enregistrer puis quitter Nano :

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print("Le programme en Python nommé ProgPython.py et écrit dans nano a été exécuté.")
```

3. Exécuter le programme (pas besoin de compilation) :

```
python ProgPython.py
```

2.5 Geany : éditeur de texte pour le développement sur Raspberry Pi

Geany est un éditeur de texte orienté développement, disponible sur Windows, macOS, Linux, et sur les Raspberry Pi équipés de processeurs ARM. Il offre une interface graphique agréable, avec notamment :

— la coloration syntaxique ;

— la prise en charge de nombreux langages (C, C++, Python, HTML, PHP, Ruby, Java...) ;

— un accès rapide à la compilation et à l'exécution.

Geany est préinstallé avec Raspberry Pi OS Desktop. Il se trouve dans le menu **Programmation**, avec d'autres outils comme Thonny Python IDE.

Exécuter du code depuis Geany

— Enregistrer le script

— Construire (Build)

— Compiler : les résultats s'affichent dans l'onglet **Compilateur** en bas de la fenêtre

— Exécuter : un terminal s'ouvre automatiquement et lance le programme

2.6 Thonny Python IDE

Thonny est un IDE (environnement de développement intégré) minimaliste conçu pour Python. Il est particulièrement adapté aux débutants. Thonny intègre son propre interpréteur Python, ce qui permet d'exécuter des scripts sans configuration préalable.

3. Utilisation des broches GPIO (entrées/sorties)

3.1 Programme Python : faire varier l'intensité d'une LED avec PWM

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Programme : LED_Python.py

import RPi.GPIO as GPIO
import time

led_pin = 18  # GPIO 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

pwm = GPIO.PWM(led_pin, 100)  # fréquence PWM de 100 Hz
pwm.start(0)  # démarrer avec 0 % de rapport cyclique

try:
    while True:
        duty = input("Rapport cyclique désiré (0 à 100) : ")
        pwm.ChangeDutyCycle(float(duty))
except KeyboardInterrupt:
    pass
finally:
    pwm.stop()
    GPIO.cleanup()
```

Exécution :

```
python3 LED_Python.py
```

3.2 Librairie PIGPIO

Installation de la librairie

Sur les dernières versions de Raspberry Pi OS, `pigpio` est préinstallée. Sinon, utilisez :

```
sudo apt-get install pigpio python3-pigpio
```

Pour démarrer le démon pigpio :

```
sudo pigpiod
```

Vérifier la version :

```
pigpiod -v
```

3.3 Programme C : faire clignoter une LED

```c
#include <pigpio.h>
#include <unistd.h> // pour sleep()

#define GPIO 14  // GPIO14 = Pin physique 8

int main(int argc, char *argv[])
{
    if (gpioInitialise() < 0) {
        return -1;
    }

    gpioSetMode(GPIO, PI_OUTPUT);

    for (int i = 0; i < 60; i++) {
        gpioWrite(GPIO, 1);
        sleep(1);
        gpioWrite(GPIO, 0);
        sleep(1);
    }

    gpioTerminate();
    return 0;
}
```

Compilation :

```
gcc -o progc progc.c -lpigpio -lrt -lpthread
```

Exécution :

```
sudo pigpiod
sudo ./progc
```

3.4 Programme Python : faire clignoter une LED

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Une LED branchée à GPIO 25 clignote toutes les 3 secondes

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)

try:
    while True:
        GPIO.output(25, GPIO.HIGH)
        time.sleep(3)
        GPIO.output(25, GPIO.LOW)
        time.sleep(3)
except KeyboardInterrupt:
    print("\nArrêt du programme.")
finally:
    GPIO.cleanup()
```

Sauvegarder ce fichier sous `blink.py` dans le répertoire `/home/pi`, puis exécuter avec :

```
cd /home/pi
python3 blink.py
```

4. Travaux Pratiques

4.1 TP 1 : Piloter les ports GPIO à partir d'un navigateur

Le but de ce TP est de piloter un port GPIO du Raspberry Pi à distance, depuis un navigateur (téléphone, tablette, PC). Pour cela, nous allons utiliser le micro-framework `web.py`, qui contient un serveur web minimaliste.

Installation du module web.py

— Installer le module :

```
python3 -m pip install web.py
```

— Si vous préférez installer web.py manuellement, voici les étapes (installer une version spécifique ou si vous n'avez pas accès à PyPI.) :

```
cd /home/pi/Téléchargements
wget https://github.com/webpy/webpy/releases/download/0.74/web.py-0.74.tar.gz
tar -xzvf web.py-0.74.tar.gz
cd web.py-0.74
sudo python3 setup.py install
```

— Installer la bibliothèque `wiringpi` si elle n'est pas présente :

```
sudo apt-get install python3-wiringpi
```

Remarque :

— Assurez-vous que le Raspberry Pi dispose de Python 3.8 ou d'une version ultérieure, car web.py 0.74 n'est pas compatible avec les versions antérieures.

— Si vous utilisez une version antérieure de Python (par exemple, 2.7), il est recommandé d'installer web.py version 0.51, qui est la dernière version compatible avec Python 2.7 .

— PyPI

— Pour vérifier la version de Python installée, vous pouvez utiliser la commande :

```
python3 --version
```

Organisation des fichiers

Créer l'arborescence suivante :

```
cd /
sudo mkdir webpyserver
cd /webpyserver
sudo mkdir templates
sudo mkdir static
```

— `webpyserver` : contient le script Python

— `templates` : contient le fichier HTML de l'interface

— `static` : pourra contenir des feuilles de style CSS

Script Python : `gpio4.py`

```python
#!/usr/bin/env python3

import web
import wiringpi
from web import form

# Initialisation du GPIO4
io = wiringpi.GPIO(wiringpi.GPIO.WPI_MODE_SYS)
io.pinMode(4, io.OUTPUT)

# Routing
urls = ('/', 'Index')
```

```
render = web.template.render('templates')

app = web.application(urls, globals())

# Formulaire
ma_forme = form.Form(
    form.Button("btn", id="btnon", value="on", html="On", class_="bouton_on"),
    form.Button("btn", id="btnoff", value="off", html="Off", class_="bouton_off")
)

class Index:
    def GET(self):
        forme = ma_forme()
        return render.index(forme, "Contrôle GPIO4 - Raspberry Pi")

    def POST(self):
        data = web.input()
        if data.btn == "on":
            io.digitalWrite(4, io.HIGH)
        elif data.btn == "off":
            io.digitalWrite(4, io.LOW)
        raise web.seeother('/')

if __name__ == "__main__":
    app.run()
```

Fichier HTML : `templates/index.html`

```
$def with (form, title)
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>$title</title>
  </head>
  <body>
    <h2>$title</h2>
    <form method="post">
      $:form.render()
    </form>
  </body>
</html>
```

Lancer le serveur Web

```
cd /webpyserver
sudo python3 gpio4.py
```

Dans un navigateur (même sur smartphone), saisir l'adresse IP du Raspberry suivie de `:8080`. Exemple : http://raspberrypi.local:8080

Remarques :

— Le port par défaut utilisé par web.py est 8080.

— Assurez-vous que le démon `pigpiod` n'est pas en conflit si vous utilisez `wiringpi`.

4.2 TP 2 : Allumer une LED avec PiGPIO en C

Objectif : Écrire un programme en C qui allume une LED pendant 10 secondes, puis l'éteint.

```
#include <pigpio.h>
#include <unistd.h> // pour sleep()

#define GPIO 17  // Remplacer par le GPIO que vous utilisez

int main()
{
    if (gpioInitialise() < 0) {
        return -1;
    }

    gpioSetMode(GPIO, PI_OUTPUT);

    gpioWrite(GPIO, 1);  // Allume la LED
```

```
    sleep(10);            // Attend 10 secondes
    gpioWrite(GPIO, 0);   // Éteint la LED

     gpioTerminate();
    return 0;
}
```

Compilation :

```
gcc -o led10s led10s.c -lpigpio -lrt -lpthread
```

Exécution :

```
sudo pigpiod   # si le démon n'est pas déjà lancé
sudo ./led10s
```