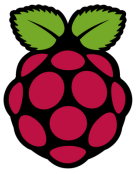
CARNUS
Enseignement supérieur RODEZ



TP Raspberry Pi

C, C++, Python ...

Prof : **Kamal Boudjelaba**

14 septembre 2025

BTS CIEL

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

IR

Table des matières

Préambule

Prérequis

- Connaissances de base de l'environnement Linux (commandes terminal).
- Notions élémentaires de programmation (structures de contrôle, variables).
- Utilisation d'un éditeur de texte en ligne de commande (comme `nano`).

Objectifs

- Savoir exécuter un programme en C, C++ ou Python sur un Raspberry Pi.
- Utiliser des outils adaptés au développement (nano, Geany, Thonny).
- Compiler et exécuter des programmes en ligne de commande.

Compétences visées

- **C04** : Analyser un système informatique
- **C08** : Coder

Activités et tâches associées

- **D1 – Élaboration et appropriation d'un cahier des charges**
  - D1-T1 : Collecte des informations
  - D1-T2 : Analyse des informations
- **D2 – Développement et validation de solutions logicielles**
  - D2-T1 : Conception de l'architecture d'une solution logicielle
  - D2-T3 : Développement, utilisation ou adaptation de composants logiciels

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.

Bilan

— L'étudiant sait utiliser des éditeurs adaptés et compiler du code sur une architecture ARM (Raspberry Pi).

— Il comprend le processus de développement logiciel dans un environnement Linux.