Synthèse des Travaux Pratiques Réseaux Informatiques

AMIRAT Samy

Sous la direction de Monsieur Morère

Master 1 MTI – Université de Lorraine

Année universitaire 2024–2025

Table des matières

Chapitre 1

Introduction

1.1 Présentation générale de la synthèse

Cette synthèse regroupe les différentes compétences acquises durant les séances de travaux pratiques du module "Réseaux informatiques et systèmes embarqués". Elle couvre trois grands axes de formation interconnectés : l'environnement UNIX, les bases pratiques des réaux informatiques, et l'exploitation du Raspberry Pi comme plateforme embarquée.

Dans une première partie, nous aborderons le système UNIX, ses commandes fondamentales, la gestion des fichiers, des répertoires, des droits d'accès, ainsi que l'utilisation d'éditeurs de texte et de scripts. Cette section comprend aussi une initiation ludique via le jeu GameShell, conçu pour renforcer la maîtrise du terminal.

La deuxième partie sera consacrée aux réaux informatiques. Nous y présenterons les fichiers de configuration réseau, les commandes utilisables en mode utilisateur et celles réservées au superutilisateur. Nous y verrons également l'utilisation de Wireshark et tcpdump pour la capture et l'analyse de trames, la configuration d'une machine virtuelle pour les tests réalistes, et la mise en place de connexions distantes via SSH.

Enfin, la troisième partie sera centrée sur le Raspberry Pi. Nous détaillerons la préparation matérielle, la configuration initiale via raspi-config, la synchronisation de l'heure, et l'utilisation de la caméra officielle. Nous explorerons aussi le contrôle d'une matrice LED via le bus I2C, la surveillance du système en ligne de commande, l'écriture de scripts bash pour l'automatisation, et le transfert de données vers une machine distante. Cette partie a été réalisée en binôme avec **Djebab Adem**.

Chacune de ces parties illustre des compétences clés pour tout futur ingénieur en systèmes embarqués et informatique industrielle. Cette synthèse vise ainsi à restituer notre expérience, nos apprentissages, nos difficultés et les solutions trouvées dans un format structuré, technique et réfléchi.

Chapitre 2

Partie UNIX et GameShell

2.1 Objectifs pédagogiques

L'objectif principal de cette première phase de travaux pratiques était de se familiariser avec l'environnement Unix/Linux en mode terminal. L'apprentissage visait à renforcer la maîtrise des commandes de base, la gestion des fichiers et des répertoires, la manipulation des permissions, ainsi que l'écriture de scripts shell simples. Le but était aussi de mieux comprendre le fonctionnement du système d'exploitation via des manipulations concrètes, tout en posant les bases nécessaires pour la suite du module orientée réseaux et Raspberry Pi.

2.2 Présentation du GameShell et son rôle dans l'initiation

GameShell est un environnement de jeu pédagogique simulé sous terminal, basé sur des principes Unix. Il a été utilisé pour introduire l'usage du shell de manière ludique. Les étudiants devaient utiliser des commandes Unix classiques pour naviguer dans une carte interactive et résoudre des énigmes en manipulant des fichiers, des droits et des processus. Ce jeu a permis d'acquérir des automatismes de commande tout en renforçant la logique de navigation et d'analyse dans un système de fichiers.

2.3 Premiers pas dans le terminal : login, shell et commandes simples

Les premières manipulations ont permis d'explorer les commandes de base telles que `pwd`, `ls`, `cd`, `clear`, `echo`, `whoami`, ou encore `exit`. Ces commandes sont fondamentales pour s'orienter dans le système de fichiers et interagir avec l'environnement de travail. L'identification de l'utilisateur courant, la reconnaissance du type de shell utilisé (bash, zsh...), ainsi que la compréhension du prompt ont été des prérequis avant d'aller plus loin.

2.4 Affichage et manipulation des variables d'environnement

Une section importante a été dédiée aux variables d'environnement comme `PATH`, `HOME`, `USER`, ou encore `PS1`. Ces variables influencent directement le comportement du terminal et des scripts. Nous avons appris à les consulter avec `echo $NOM_VARIABLE`, à les modifier temporairement (`export VAR=valeur`), et à comprendre leur rôle dans l'exécution des commandes et des programmes.

2.5 Création, suppression et manipulation de fichiers/-répertoires

Les commandes `touch`, `mkdir`, `rmdir`, `rm`, `cp`, et `mv` ont été utilisées pour créer, déplacer, copier, renommer ou supprimer des fichiers et des dossiers. Nous avons aussi appris à utiliser les options comme `-r` (récursif), `-f` (force), ou `-i` (confirmation interactive). Ces manipulations simples sont essentielles dans tout environnement Unix.

2.6 Arborescence et navigation dans le système de fichiers

L'arborescence Unix suit une structure hiérarchique avec une racine `/`, des dossiers systèmes (`/bin`, `/etc`, `/home`, `/var`, etc.) et des répertoires utilisateurs. Grâce aux commandes `tree`, `ls -R` ou `find`, nous avons visualisé cette organisation et appris à nous déplacer efficacement grâce à des chemins relatifs (`../`, `./`) ou absolus (`/home/user`).

2.7 Permissions, droits d'accès et gestion des utilisateurs

Les permissions sous Unix sont basées sur un modèle utilisateur/groupe/autres, avec les droits `r`, `w`, `x`. Grâce à la commande `ls -l`, nous avons analysé les droits des fichiers, et modifié ceux-ci avec `chmod`, `chown`, et `chgrp`. Ces notions sont essentielles pour la sécurité, l'administration du système et les scripts.

2.8 Utilisation des jokers et redirections

Les jokers (`*`, `?`, `[]`) ont été utilisés pour manipuler plusieurs fichiers en une seule commande. Les redirections (`>`, `>>`, `<`, `|`) ont permis de sauvegarder la sortie d'une commande dans un fichier ou de chaîner plusieurs commandes avec `pipe`. Ces éléments sont essentiels pour automatiser des tâches ou manipuler des flux de données.

2.9 Utilisation des éditeurs de texte (vim, nano, gedit)

L'édition de fichiers est incontournable sous Unix. Nous avons utilisé des éditeurs en ligne de commande comme `nano` (intuitif) et `vim` (plus puissant mais complexe), ainsi que

des éditeurs graphiques comme `gedit`. L'édition de fichiers de configuration, de scripts ou de notes a été intégrée dans plusieurs TPs.

2.10 Gestion des processus et scripts shell de base

Nous avons découvert la gestion des processus avec `ps`, `top`, `kill`, `jobs`, et l'exécution en arrière-plan (`&`). L'écriture de scripts en bash a été abordée avec des fichiers `.sh` contenant des instructions simples : affichage de texte, boucles, tests conditionnels, et sauvegardes automatiques. Un exemple typique : sauvegarder le répertoire `Documents` dans un dossier daté automatiquement.

Chapitre 3

Partie Réseaux

3.1 Objectifs de l'apprentissage

Cette partie des travaux pratiques visait à explorer et comprendre les outils de configuration et de diagnostic réseau sous Linux. L'apprentissage s'est appuyé sur des fichiers système, des commandes fondamentales, l'analyse réseau à l'aide de Wireshark, et la mise en place d'une machine virtuelle Linux afin de dépasser les limitations du mode utilisateur classique.

3.2 Fichiers de configuration réseau

Nous avons analysé plusieurs fichiers essentiels au fonctionnement réseau d'un système Linux :

- `/etc/hosts` : pour la résolution de noms locale ;
- `/etc/resolv.conf` : liste les serveurs DNS utilisés ;
- `/etc/hostname` : contient le nom de la machine ;
- `/etc/network/interfaces` : pour la configuration manuelle des interfaces ;
- `/etc/services` et `/etc/protocols` : pour l'association des ports/services.

Ces fichiers sont souvent modifiés par l'administrateur système pour adapter la connectivité et les services en fonction du réseau cible.

3.3 Commandes utilisables en mode utilisateur

En tant qu'utilisateur standard (non-root), plusieurs commandes réseau restent disponibles :

- `ping <adresse>` : tester la connectivité IP ;
- `ip a` ou `ifconfig` : afficher les interfaces réseau ;
- `wget <url>` : télécharger un fichier distant ;
- `traceroute <ip>` : suivre le chemin des paquets ;
- `nslookup <domaine>` : interroger un serveur DNS ;
- `ssh user@ip` : connexion distante via SSH si le serveur est actif.

Ces outils ont permis un premier diagnostic de notre réseau local.

3.4 Commandes nécessitant les droits administrateurs

Certaines commandes essentielles ne fonctionnent qu'en mode superutilisateur :
— `tcpdump -i interface` : capture des paquets en temps réel ;
— `wireshark` : outil graphique puissant d'analyse réseau ;
— `nmap` : scanner de ports ;
— `systemctl restart ssh` : gestion des services réseau ;
— `ip link set <interface> up/down` : gestion des interfaces.
Ces commandes étant bloquées sur les machines de la salle, nous avons dû recourir à une solution externe.

3.5 Mise en place de la machine virtuelle

La machine virtuelle (VM) s'est révélée indispensable pour dépasser les limitations du mode utilisateur. Nous avons installé VirtualBox sur nos postes personnels, puis configuré deux machines virtuelles Debian : une en tant que client et l'autre en tant que serveur.

Les avantages obtenus avec ce setup :
— Possibilité de se connecter en administrateur (`sudo su`) ;
— Installation de tous les paquets réseau nécessaires ;
— Captures complètes avec `tcpdump` et Wireshark ;
— Tests réseau entre client et serveur en réseau interne VirtualBox.
Cette solution nous a permis de valider l'ensemble des manipulations prévues dans le TP, sans compromettre la sécurité du système hôte.

3.6 Connexion distante et redirection de ports SSH

Le protocole SSH permet une connexion sécurisée à distance entre deux machines. Pour tester la redirection de port, nous avons exécuté :

```
ssh machine-passerelle -L 9000:192.168.1.254:80
```

Cela crée un tunnel sécurisé permettant d'accéder à un service HTTP distant via `localhost:9000`. Cette manipulation est utile pour accéder à des services protégés derrière un pare-feu.

3.7 Wireshark et tcpdump : analyse des trames réseau

Wireshark est un outil graphique permettant de visualiser les paquets échangés sur le réseau. Il permet d'appliquer des filtres comme :
— `http.request.full_uri contains "wikipedia.org"` ;
— `tcp.port == 22` (trafic SSH) ;
— `icmp` pour filtrer les pings.
L'alternative en ligne de commande est `tcpdump`, par exemple :

```
tcpdump -i eth0 -nn -v
```

Cette commande permet de capturer le trafic brut. Nous avons redirigé la sortie vers un fichier pour l'analyser ensuite dans Wireshark.

3.8 Topologie réseau de notre environnement VirtualBox

Pour représenter concrètement notre environnement de test, nous avons mis en place deux machines virtuelles sous Debian, configurées comme un client et un serveur, toutes deux hébergées sur une machine réelle à l'aide de VirtualBox. La machine réelle était connectée à Internet via Wi-Fi, tandis que les deux VM communiquaient entre elles dans un réseau interne isolé (réseau privé VirtualBox).

Chaque machine virtuelle possédait une adresse IP distincte dans la plage 192.168.56.X, attribuée automatiquement par VirtualBox en mode « réseau interne ». Ce type de configuration nous a permis de simuler un vrai réseau local sans aucune dépendance au réseau de l'université, ce qui était essentiel pour tester les échanges SSH, les scans de ports, les transferts de fichiers ou les captures de trames via tcpdump.

Le serveur Debian jouait le rôle de passerelle ou de cible à analyser, et le client Debian servait de poste d'analyse. L'ensemble a constitué un environnement sécurisé, réaliste et facilement reproductible pour l'ensemble des expérimentations réseau du TP.

3.9 Bilan de la partie Réseaux

Ce TP nous a permis de mieux comprendre l'infrastructure réseau sous Linux, les outils d'analyse, les limitations des droits utilisateurs, et l'utilité d'une machine virtuelle pour simuler des scénarios avancés. Cette approche pédagogique, très pratique, a enrichi notre compréhension des concepts réseaux.

Chapitre 4

Partie Raspberry Pi

4.1 Objectifs et présentation du Raspberry Pi

Cette dernière partie du module a été réalisée en binôme avec **Djebab Adem**. L'objectif principal était de découvrir le fonctionnement et la configuration d'un système Linux embarqué à travers l'utilisation d'un **Raspberry Pi**. Ce micro-ordinateur à bas coût nous a permis de manipuler un environnement complet, d'accéder à distance au système, d'utiliser des périphériques matériels comme la caméra ou une matrice LED, et de développer des scripts simples pour automatiser des tâches.

4.2 Préparation matérielle et démarrage de la carte

Nous avons commencé par préparer une carte microSD contenant l'image officielle du système Raspbian. La carte a été insérée dans le Raspberry Pi 3B+, puis ce dernier a été branché à un écran via HDMI, à un clavier et une souris en USB, et à l'alimentation. Une fois le système lancé, l'écran de connexion est apparu avec les identifiants par défaut (`pi`/`raspberry`). Cette étape était cruciale pour valider le bon démarrage et accéder à l'environnement Linux.

4.3 Configuration système locale et outils intégrés

Après le premier démarrage, nous avons lancé `raspi-config` pour ajuster les paramètres de base du système :

- Configuration de la langue (français), du clavier (azerty) et du fuseau horaire (Europe/Paris).
- Activation de la caméra et du bus I2C dans le menu `Interface Options`.
- Connexion au Wi-Fi et configuration de la localisation réseau.

D'autres modifications ont été faites dans le fichier `/boot/config.txt` pour affiner les réglages. Le tout a été validé par un redémarrage avec `sudo reboot`.

4.4 Synchronisation de l'heure et gestion NTP

Le Raspberry Pi ne dispose pas d'horloge temps réel (RTC). Sans accès Internet, l'heure système est incorrecte. Pour corriger cela, nous avons utilisé :

— `sudo apt install ntpdate`
— `sudo service ntp stop`
— `sudo ntpdate pool.ntp.org` (mise à jour de l'heure)
— `sudo service ntp start`

Cette procédure assure une synchronisation fiable à chaque démarrage dès que le Raspberry Pi est connecté à Internet.

4.5 Contrôle d'une matrice LED avec le bus I2C

Dans cette étape du TP, nous avons utilisé le bus I2C du Raspberry Pi pour communiquer avec une matrice LED 8x8. Le protocole I2C (Inter-Integrated Circuit) permet à plusieurs périphériques de dialoguer avec un microcontrôleur. Après avoir activé le bus dans `raspi-config`, nous avons installé les outils nécessaires et vérifié la connexion avec `i2cdetect -y 1`.

Plusieurs scripts Python nous ont été fournis. Le premier (11.2.1) allumait chaque LED une par une, créant un effet fluide. Le second (11.2.2) affichait une phrase défilante comme *"Sir... on en a gros !!!"*. Un autre script (11.2.4) permettait d'afficher une image ASCII en transformant des caractères en couleurs. Enfin, l'activité 11.2.5 proposait une animation style Pac-Man avec plusieurs images en séquence. Ce travail a montré comment piloter un affichage LED avec Python et le Raspberry, et a illustré la souplesse du protocole I2C pour des affichages embarqués.

4.6 Surveillance système avec le Raspberry Pi

Cette activité portait sur la récupération d'informations système internes (température, fréquence CPU, etc.). Par exemple :

— `cat /sys/class/thermal/thermal_zone0/temp` retourne la température en millidegrés.
— `top`, `htop`, `free -h` permettent de suivre les performances.

Nous avons converti la température pour un affichage lisible, et compris que Linux donne un accès direct à l'état de la machine via le système de fichiers `/proc` et `/sys`. Ces outils sont utiles pour les projets embarqués sensibles à la chaleur ou à la charge système.

4.7 Prise de vue avec le module caméra

Nous avons utilisé le module caméra officiel du Raspberry Pi. Pour capturer une image :

```
sudo raspistill -o image.jpg
```

Nous avons testé les options `-width`, `-height`, `-quality` pour ajuster la taille et la compression. Différents formats ont été comparés (JPG rapide, PNG lent, BMP intermédiaire). L'option `-timelapse` permettait de capturer automatiquement plusieurs images à intervalle régulier. Cette fonctionnalité ouvre des applications intéressantes comme le suivi d'une plante ou la météo.

4.8 Création d'un utilisateur dédié et gestion des droits

Dans une optique de gestion plus propre, nous avons créé un nouvel utilisateur pour les opérations de capture :

```
sudo adduser capture
sudo usermod -aG video capture
```

L'ajout au groupe `video` était nécessaire pour accéder à la caméra. Cette approche renforce la sécurité du système en limitant les droits.

4.9 Transfert de fichiers (images ou scripts)

Nous avons utilisé `scp` pour transférer les images depuis le Raspberry Pi vers notre PC local. Exemple :

```
scp pi@192.168.X.X:image.jpg ./
```

Ce transfert sécurisé par SSH a facilité la récupération et l'analyse des résultats sur nos machines personnelles.

4.10 Automatisation de la capture avec script Bash

Nous avons écrit un script pour automatiser les prises de vue et rendre le système plus autonome :

```
#!/bin/bash
while true; do
  filename="photo_$(date +%Y%m%d_%H%M%S).jpg"
  raspistill -o $filename
  sleep 30
done
```

Ce script réalise des captures toutes les 30 secondes, utile pour des projets de time-lapse ou de la surveillance. Nous avons testé ce script avec succès en mode terminal

4.11 Difficultés rencontrées et travail en binôme

Le principal défi a été la configuration réseau, parfois instable en Wi-Fi. Certains réglages (clavier, caméra, NTP) nécessitaient de redémarrer plusieurs fois le système. Grâce au travail en binôme avec **Djebab Adem** , nous avons pu nous répartir les tâches, identifier plus vite les erreurs (permissions, chemins, accès caméra), et documenter chaque étape du TP

4.12 Analyse pédagogique et conclusion

Ce TP nous a permis d'expérimenter concrètement l'administration système sur une plateforme embarquée. Nous avons configuré le système, utilisé des capteurs et périphériques (caméra, LED), automatisé des tâches, et surveillé les performances. Le Raspberry

Pi est un outil pédagogique puissant pour initier aux systèmes embarqués et aux projets IoT.

Chapitre 5

Conclusion générale de la synthèse

Au terme de cette synthèse, réalisée à partir des différentes séances de travaux pratiques, nous avons pu explorer trois axes fondamentaux de la formation : les bases du système UNIX, les notions pratiques en réseaux informatiques, et l'exploitation du Raspberry Pi comme plateforme embarquée.

Les premières séances consacrées à UNIX nous ont permis de mieux comprendre le fonctionnement du terminal, la structure des fichiers, la gestion des utilisateurs et des permissions, ainsi que l'importance de l'automatisation via des scripts bash. Le jeu GameShell a constitué une introduction originale et ludique à cet environnement.

La deuxième partie, orientée réseaux, nous a amenés à utiliser de nombreuses commandes de diagnostic, comprendre les privilèges liés au superutilisateur, analyser des trames avec Wireshark et tcpdump, et configurer un environnement virtualisé complet pour simuler des scénarios réalistes. L'utilisation d'une machine virtuelle a permis d'expérimenter sans contrainte les outils réseau.

Enfin, la troisième partie sur le Raspberry Pi a concrétisé les notions précédentes dans un cadre embarqué. Nous avons pu configurer un système distant, manipuler du matériel via les interfaces GPIO/I2C, interagir avec une matrice LED, utiliser un module caméra, et même automatiser des actions à l'aide de scripts. Le travail en binôme avec **Djebab Adem** a renforcé notre rigueur et notre autonomie.

Cette synthèse nous a permis d'enrichir notre compétence en administration Linux, en communication réseau, et en mise en œuvre de systèmes embarqués. Elle constitue une base solide pour aborder des projets plus complexes en informatique industrielle, supervision, ou Internet des objets (IoT).