



# Catalogue des formations

Avril-Juin 2010

25, avenue de l'hôtel de ville  
91130 – Ris -Orangis

<http://www.logilin.fr>

contact : [formations@logilin.fr](mailto:formations@logilin.fr)

## Avant-Propos



Nous sommes spécialistes en formations :

- intra-entreprises (dans vos locaux ou dans les nôtres)
- individuelles accélérées : le programme est adapté pour convenir aux impératifs du stagiaire.

Tous nos cours sont des créations originales de Logilin, conçus ainsi que les travaux pratiques accompagnant les sessions par nos intervenants.

Nous vous proposons deux thèmes principaux de formations, chacun regroupant différentes filières au sein desquelles des formations de niveaux croissants permettent d'acquérir une véritable expertise.

- Thème « Développement »
  - ◆ Filière « Concepts et Méthodes »
  - ◆ Filières « Langages »
- Thème « Linux »
  - ◆ Filière « Utilisation »
  - ◆ Filière « Administration »
  - ◆ Filière « Linux Industriel »

## PREMIÈRE PARTIE

### Thème « Développement »

#### *Filière « Méthodes de programmation »*

- Introduction à la programmation
- Maîtriser Eclipse et le CDT (*sur demande*)

#### *Filière « Langages de programmation »*

- Programmer en Langage C
- Programmer en C++
- Programmer en Langage PHP
- Développer en Python (*sur demande*)
- Développer avec Ruby on Rails (prévu pour avril 2009)
- C embarqué pour micro-contrôleur

Ce cours présente les concepts et les méthodes de programmation utilisés pour le développement d'applications, la création de sites web, l'interrogation de bases de données, etc. Les notions d'algorithmes et de structuration des données sont abordées, et les rudiments de plusieurs langages de programmation sont proposés.

## Organisation

---

**Durée** : 4 jours (28 heures).

**Pré-requis** : aucun.

**Conseil cursus** : Pour prolonger cette formation, nous vous proposons ensuite plusieurs cours à choisir en fonction de votre environnement de travail : la formation « *Programmer en Langage C* » intéressera la plupart des professionnels du monde industriel, la formation « *Programmer en PHP* » vous permettra de développer des sites web dynamiques performants et sûrs, et enfin la formation « *Ecriture de Scripts Shells* » étendra les compétences des administrateurs système Unix ou Linux.

## Thèmes abordés

---

- Notions principales : composants d'un ordinateur, instructions et données, séquences et algorithmes.
- Les principaux langages actuels : types de langages, la famille C, C++, Java, le shell Unix et ses scripts, les langages Perl, PHP, Ruby, bases de données et requêtes SQL, Visual Basic et C#.
- Algorithmes : structures de contrôles, représentation en pseudo-code, jeu d'instructions
- Les variables : représentation en mémoire, types, exemples dans divers langages
- Les structures de contrôle : les tests, les itérations, les procédures
- Les bibliothèques de fonctions : nécessité des bibliothèques, bibliothèques utilisateur et bibliothèque système.
- Le langage C : la syntaxe globale, les mots-clés du langage, les types de variables, quelques fonctions de la bibliothèque C.
- Programmation orientée objet : concepts, définition de classes, instanciation, exemples en C et C++.
- Programmation événementielle graphique : concepts, bibliothèques de contrôles, exemples en Visual Basic.
- Langages pour le Web : les balises HTML, les scripts Javascript, le langage PHP
- Les bases de données : concepts, les requêtes SQL, création d'une base et d'une table, insertion d'enregistrements, recherches, modifications et suppressions.

## Travaux pratiques

---

Les exemples de ce cours sont présentés par l'intervenant dans les environnements Unix/Linux et Windows. Les participants peuvent aborder leurs travaux pratiques sur l'environnement de leur choix.

Plan détaillé au verso ►

## Plan détaillé

---

### Notions principales

#### Composants d'un ordinateur

Unité centrale, processeur, mémoire, disques, mémoire volatile et non volatile.

#### Instructions et données

Éléments manipulés par un processeur, données binaires, jeu d'instructions élémentaires, assembleur, notion de compilation, fichiers source et exécutable.

#### Séquences et algorithmes

Organisation des opérations, séquençement des actions, nécessité des algorithmes.

### Les principaux langages actuels

#### Types de langages

Langages compilés et interprétés, scripts, requêtes système et bases de données.

#### La famille C/C++/Java

Caractéristiques des langages de la famille C, aperçu de codes source C, C++, Java, exemples de compilation et d'exécution.

#### Le shell Unix et ses scripts

Principe du shell Unix et des scripts, exemples avec différents interpréteurs.

#### Les langages Perl / PHP / Ruby

Scripts de haut-niveau, avantages, inconvénients, utilisations, exemples de scripts employés dans des pages web dynamiques.

#### Bases de données et requêtes SQL

Les bases de données actuelles, SGBD disponibles, langage de requête SQL.

#### Visual Basic et C#

Concept de programmation événementielle graphique, environnements Visual Studio Express sous Windows, exemple de programme en Visual Basic.

### Algorithmes

#### Structures de contrôles

Structures de contrôles indispensables : les boucles, les tests, les sous-programmes.

#### Représentation en pseudo-code

Écriture d'algorithme, notion de pseudo-code source.

#### Jeu d'instructions

Itération, de sélection, appels de sous-routines, affectations de variables, etc.

### Les variables

#### Représentation en mémoire

Représentation binaire des données, écriture hexadécimale, stockage de données entières, réelles, alphabétique, etc.

#### Types

Les types scalaires, les tableaux, les chaînes, les structures.

#### Exemples dans divers langages

Déclarations en C, en Visual Basic, en Perl. Dans les langages interprétés.

### Les structures de contrôle

#### Les tests

Tests conditionnels, opérateurs booléens, conditions logiques.

#### Les itérations

Boucle conditionnelles et itératives, performances et optimisation.

#### Les procédures

Découplage des fonctionnalités, appels-multiples, passage de paramètres, fonctions et valeurs de retour, variables locales et globales.

## Les bibliothèques de fonctions

### Nécessité des bibliothèques

Séparations entre langage et fonctionnalités système, partage entre applications.

### Bibliothèques utilisateur et bibliothèque système

Création de bibliothèque, réutilisabilité, compilation séparée, optimisation.

## Le langage C

### Syntaxe globale, mots-clés du langage et types de variables

Exemples de code, détails de syntaxe, compilation. Présentation des structures de contrôle, des variables et des fonctions.

### Aperçu des pointeurs

Déclaration et utilisation des pointeurs sur des entiers, des caractères, chaînes.

### Quelques fonctions de la bibliothèque C

Principe des fichiers d'entête, inclusion, édition des liens, bibliothèque d'entrée-sorties standards, manipulation des chaînes de caractères, etc.

## Programmation orientée objet

### Concepts

Classes et instances, attributs et méthodes, encapsulation, réutilisabilité, héritage.

### Définition de classes et instanciation

Classes en C, C++, Java, PHP, création de nouveaux objets, dérivation de classes.

### Exemple en langage C

Concept de structures, pointeurs de fonctions, compilation séparée et encapsulation.

## Programmation événementielle graphique

### Concepts

Composants graphiques, événement fournis par le système d'exploitation.

### Bibliothèques de contrôles

Utilisation de Visual Studio Express, conception de feuilles, insertion de code.

### Exemples en Visual Basic

Création d'un outil de conversion de mesures entre systèmes métrique et anglais.

## Langages pour le Web

### Les balises HTML

Principe, entités et balises courantes, création manuelle d'une page web valide.

### Les scripts Javascript

Insertion de code Javascript dans une page HTML, limites de l'utilisation.

### Le langage PHP

Principe, inclusion du code dans une page HTML, aperçu de la syntaxe, exemples de code pour page web dynamique.

## Les bases de données

### Concepts

Persistance des données, stockage, fichiers, bases de données relationnelles.

### Les requêtes SQL

Langage d'interrogation, principe, intégration dans d'autres langages.

### Exemples d'accès à une base de données

Création de table, insertion et recherche d'éléments, modifications et suppressions.

## Conclusion

Discussions et expérimentations libres sur l'ensemble des thèmes abordés.

Conçu dans les années soixante-dix, le langage C reste encore de nos jours l'un des piliers de la programmation dans de nombreux domaines : industriel, scientifique, réseau, bas-niveau, etc.

Puissant, efficace, épuré, le C est à l'origine de nombreux langages dérivés (C++, java, C#...), et une bonne connaissance de ce langage est un atout réel pour la maîtrise du développement logiciel.

### Organisation

---

**Durée** : 4 jours (28 heures).

**Pré-requis** : le stagiaire doit avoir des notions générales de programmation.

**Conseil cursus** : en préambule à cette formation, nous vous conseillons notre formation « Introduction à la Programmation ».

### Thèmes abordés

---

- **Concepts du langage C** : présentation, avantages et inconvénients, utilisations typiques.
- **Outils de développement** : environnement, compilateur, débogueur.
- **Structures des programmes** : exemple, vocabulaire, fonctions et variables, constantes, types et expressions.
- **Les variables du C** : globales et locales, détail des types scalaires, tableaux.
- **Les fonctions** : passage d'arguments, types de retour, passage par valeur, passage par référence.
- **Les pointeurs** : déclarations, pièges, arithmétique des pointeurs, initialisation et utilisation.
- **Structures de contrôle** : tests et itérations, sélections.
- **Chaînes de caractères** : pointeurs et tableaux de caractères, fonctions standard de manipulation de chaînes.
- **Expressions du C** : opérateurs mathématiques, logiques et binaires.
- **Structures de données** : principes et déclaration, utilisation, tableaux de structures, pointeurs de fonction, programmation objet.
- **Allocation dynamique de mémoire** : allocation et libération, organisation des données.

### Travaux pratiques

---

Les travaux pratiques de ce cours peuvent se dérouler, au choix du stagiaire, sur système Linux avec Eclipse/CDT ou sur système Windows avec Visual Studio C++.

*Plan détaillé au verso ►*

## Plan détaillé

---

### Concepts du langage C

#### Présentation

Premières versions du langage C, évolutions, standards et normes

#### Avantages et inconvénients

Efficacité et performance, proximité du matériel et du processeur, portabilité, laxisme du langage, exemples de bogues courants et des conséquences désastreuses.

#### Utilisations typiques

Systèmes industriels et embarqués, serveurs et réseaux, télécommunication, gestion de données.

### Outils de développement

#### Environnements

Présentation de l'environnement de développement, prise en main d'Eclipse sous Linux ou Visual C++ sous Windows

#### Compilation

Création d'un projet et compilation d'un premier programme, exécution et résultat. Rôle de la bibliothèque C.

#### Débogage

Exécution du programme en pas à pas, points d'arrêts.

### Structures des programmes

#### Aspect et vocabulaire du C

Présentation de code, indentation, caractères spéciaux, mots-clés, définition et déclaration de fonctions, fichier d'entête, fonctions de bibliothèque.

#### Fonctions

Ecriture des fonctions, différences entre déclaration et définition, paramètres.

#### Variables

Types des variables, déclaration et utilisation, expression et constantes.

### Les variables du C

#### Portée et persistance

Stockage des variables en pile, variables globales et variables locales, variables automatiques et statiques.

#### Les types scalaires

Format et représentation interne, utilisation.

#### Les tableaux

Tableaux d'entiers, déclaration, initialisation et utilisation.

### Les fonctions

#### Passage d'arguments

Utilisation de la pile, types des paramètres, modification des paramètres formels, passage par valeur, réservation de l'espace dans la pile.

#### Types de retour

Utilisation des valeurs de retour des fonctions. Limitation à une valeur.

#### Passage d'argument par référence

Nécessité de passer un pointeur dans la pile, modification des paramètres d'appel.

### Les pointeurs

#### Déclarations des pointeurs

Exemple de déclaration des types, affichage des valeurs des pointeurs et des contenus pointés. Risques liés à l'utilisation des pointeurs.

#### Arithmétique des pointeurs

Concepts, vérification des pas d'incrémentations en fonction des types de données.

## Structures de contrôle

### Structures de test

Structure if / else. Imbrications et indentation. Expression testée. Confusion entre égalité et affectation.

### Structure de sélection

Construction switch / case. Utilisation du break.

### Itération conditionnelle

Boucles while et until. Boucle infinie. Rupture de séquence.

### Enumération

Boucle for. Syntaxe usuelle et variantes. Rupture de séquence.

## Chaînes de caractères

### Principes des chaînes en C

Pointeurs et tableaux de caractères, représentation, caractère nul.

### Fonctions standards de la bibliothèque C

Longueur, copie, concaténation, recherche de sous-chaîne.

## Expressions du C

### Opérateurs arithmétique

Opérateurs classiques, notations condensées, pré- ou post- incrémentation et décrémentation.

### Opérateurs logiques

Valeurs de vérité, composition d'expressions booléennes

### Opérateurs binaires

ET, OU, OU exclusif, négation, décalage, masques

## Structures de données

### Principes

Utilités des structures, déclaration, et utilisation des structures

### Tableaux

Tableaux et pointeurs de structures, initialisation et utilisation.

### Programmation objet

Pointeurs de fonction, encapsulation, attributs et méthodes, conception orientée objet.

## Allocation dynamique de mémoire

### Principes

Utilité des allocations dynamiques, fonctions d'allocation et de libération.

### Fuites mémoire

Risques des allocations, règles de bonne conduite, outils de débogage

### Structures de données dynamique

Implémentation des listes chaînées et doublement chaînées, utilisation.

## Conclusion

Discussions et expérimentations libres sur l'ensemble des thèmes abordés.

Le langage C++ est le principal descendant du langage C datant des années 1970. Améliorant la robustesse de ce dernier, et intégrant totalement les préceptes de la programmation orientée objet, le langage C++ est devenu un standard incontournable du développement logiciel actuel. Vous découvrirez durant cette formation tous les apports du C++ et de la programmation objet.

### Organisation

---

**Durée** : 5 jours (35 heures).

**Pré-requis** : le stagiaire doit avoir des notions générales de programmation et une connaissance préliminaire du C est appréciable.

**Conseil cursus** : en préambule à cette formation, nous vous conseillons notre formation « Introduction à la Programmation » et le cours « programmer en langage C ».

### Thèmes abordés

---

- **Introduction** : C et C++, avantages et inconvénients et évolutions, perspectives.
- **Outils et méthodes de développement** : compilateurs, environnement de développement, débogage.
- **Rappels sur le C** : syntaxe, types, structures de contrôles, compilation séparée.
- **Spécificités syntaxiques du C++** : éléments syntaxiques, références, fonctions, structures de données, types de données.
- **Gestion des exceptions** : principes, propagation.
- **Programmation orientée objet** : principes, encapsulation, héritage.
- **Objets en C++** : classes, encapsulation, composition de classes, polymorphisme.
- **Les templates** : principe, *templates* de fonctions, *templates* de classes.
- **Bibliothèques du C++** : *streams*, *Standard Template Library*.

### Travaux pratiques

---

Les travaux pratiques de ce cours peuvent se dérouler, au choix du stagiaire, sur système Linux avec Eclipse/CDT ou sur système Windows avec Visual Studio C++.

*Plan détaillé au verso ►*

## Plan détaillé

---

### Introduction

#### Le C et le C++

Historique, présentation, relations entre C et C++.

#### Avantages et Inconvénients

Robustesse, réutilisabilité, limites, standards.

#### Évolutions et perspectives

Langages concurrents et dérivés, *Java*, *C#*, etc.

### Outils et méthodes de développement

#### Compilateurs

*Gnu G++*, *Visual C++*, *Borland C++ builder*, *Intel C++ Compiler*.

#### Environnement de développement

*Eclipse* et *C/C++DT*, *Visual Studio*.

#### Débogage

*GDB*, *Code View* et *Visual Studio*.

### Rappels sur le langage C

#### Syntaxe élémentaire

Éléments syntaxiques, déclarations et définitions, commentaires, préprocesseur.

#### Types de données

Types scalaires et structures, constantes, pointeurs.

#### Structures de contrôles

Tests et boucles, ruptures de séquence, fonctions, arguments.

#### Compilation séparée

Prototypes, fichiers d'en-tête, modules, *makefile* etc.

### Spécificités syntaxiques du C++

#### Éléments syntaxiques

Commentaires, surcharge des opérateurs, espaces de nom, opérateur `::`.

#### Références

Exemples, relation entre pointeurs et références, ramasse-miettes.

#### Les fonctions

Paramètres par défaut, fonctions *inline*, fonctions génériques.

#### Structures de données

Initialisation, opérateur d'allocation et libération.

#### Types de données

Constantes, booléens, identification dynamique (*run-time*).

### Gestion des exceptions

#### Principes

Déclenchement d'exception, bloc *try/catch*.

#### Propagation

Remontée d'exceptions.

### La programmation orientée objet

#### Principes

Classes et instances, membres et méthodes, intérêts.

#### Encapsulation

Protection, accesseurs, constructeurs.

#### Héritage

Dérivation de classe et spécialisation.

## Les objets du C++

### **Classes**

Définition, membres, méthodes, argument *this*, constructeur et destructeur, copie.

### **Encapsulation**

Accesseurs, membres publics, privés et protégés, fonctions amies.

### **Composition de classes**

Association, héritage simple, dérivation, héritage multiple.

### **Polymorphisme**

Méthodes virtuelles, structures polymorphes, classes abstraites.

## Les templates

### **Principe**

Programmation générique, efficacité, exemple.

### **Templates de fonctions**

Exemple, utilité.

### **Templates de classes**

Méthode de *template*, instantiation.

## Bibliothèques du C++

### **Entrées-sorties *stream***

Flux, opérateurs, surcharge.

### ***Standard Template Library***

Conteneurs, itérateurs, recherches et tris.

## Conclusion

Discussions et expérimentations libres sur l'ensemble des thèmes abordés.

*Libre et performant, le langage PHP s'incorpore à merveille dans les pages Web dynamiques, et il est au coeur de la plupart des grands sites Internet professionnels. Si le langage est simple à aborder et à utiliser, il est néanmoins difficile d'appréhender la richesse de sa bibliothèque, ainsi que les précautions subtiles à prendre pour éviter toute faille de sécurité.*

*Ce cours vous permettra d'écrire des scripts robustes, lisibles et sûrs, autant pour des sites Web que pour des applications locales indépendantes.*

### Organisation

---

**Durée** : 3 jours (20 heures).

**Pré-requis** : le stagiaire doit avoir des notions générales de programmation.

**Conseil cursus** : en préambule à cette formation, nous vous conseillons notre formation « Introduction à la Programmation ». Pour approfondir vos compétences en administration d'un serveur Web nous vous proposons ensuite notre formation « Administration d'un serveur LAMP ».

### Thèmes abordés

---

- Présentation de PHP : concepts, avantages et inconvénients, concurrents, évolutions.
- Prise en main : installation d'Apache et de PHP. Interpréteur.
- Principes du langage : incorporation dans les fichiers HTML, exemples simples, syntaxe du langage.
- Les variables de PHP : scalaires, tableaux, prédéfinies.
- Structures de contrôle : itérations, sélection.
- Fonctions et opérateurs : fonctions utilisateurs, fonctions de bibliothèques, opérateurs mathématiques et logiques.
- Programmer Objet en PHP : classes et instances, attributs et méthodes.
- Les formulaires HTML : définition d'un formulaire, récupération des informations, méthodes de transmission de données entre pages.
- Les sessions : authentification, sessions et cookies. Sécurité d'un script PHP.
- Les fichiers : ouverture et accès aux fichiers, transferts de données.
- Bases de données : accès à la base, requêtes SQL.

### Travaux pratiques

---

Les travaux pratiques de ce cours s'appuient sur des serveurs LAMP (Linux, Apache, MySQL, PHP) et visent à mettre en oeuvre un site web dynamique fonctionnant sur le modèle des grands sites commerciaux.

*Plan détaillé au verso ►*

## Plan détaillé

---

### Présentation de PHP

#### Concepts

Principe des langages de scripts, nécessité d'un web dynamique, héritage (C, Perl...)

#### Avantages et inconvénients

Performances et limites, qualité de code et facilité de maintenance.

#### Concurrents et évolutions

Positionnement face à d'autres langages comme ASP, Python ou Ruby on Rails.

### Prise en main

#### Serveur HTTP

Installation d'Apache et de PHP sur un serveur Linux.

#### Interpréteur

Installation et test de l'interpréteur en ligne de commande, premiers exemples de scripts PHP.

### Principes du langage

#### Intégration dans HTML

Aperçu du langage HTML, possibilités d'incorporer du code PHP, efficacité des différentes méthodes.

#### Syntaxe du langage

Écriture de scripts simples, syntaxe, caractères spéciaux, inclusion de fichiers.

### Les variables de PHP

#### Les variables scalaires

Déclaration et utilisation, conversion entre types, équivalence

#### Les tableaux

Initialisation et utilisation, passage en argument de fonctions

#### Variables prédéfinies

`$GLOBALS`, `$_SERVER`, `$_GET`, `$_POST`, `$_SESSION`...

### Structures de contrôle

#### Structures itératives

Boucles while, for et foreach. Rupture de séquence.

#### Structures sélectives

Test If, Switch..

### Fonctions et opérateurs

#### Fonctions utilisateurs

Définition, utilisation, arguments, passage par valeur ou référence, nombre variables, valeur de retour.

#### Fonctions de bibliothèques

Extensions, fonctions internes.

#### Opérateurs

Arithmétiques, binaires et logiques. Opérateurs sur les chaînes et les tableaux.

### Programmer Objet en PHP

#### Classes et instances

Déclaration, héritage, abstraction/

#### Attributs

Visibilité, constantes.

#### Méthodes

Constructeurs, destructeurs, surcharge.

### Les formulaires HTML

#### Définition d'un formulaire

Eléments des formulaires HTML, initialisation des contenus

### **Récupération des informations**

Méthodes de transmissions de données, traitement dans le même script ou sur une autre page.

## **Les sessions et la sécurité**

### **Mécanismes d'authentification**

Authentification simple, cookies, sessions.

### **Sécurité d'un script PHP**

Les risques liés à l'enchaînement des scripts, à la transmission de données.

## **Les fichiers**

### **Accès aux fichiers**

Ouverture, lecture et sauvegarde,

### **Accès aux fichiers distants**

Problèmes de sécurités. Téléchargements.

## **Bases de données**

### **Principes**

Utilité d'un SGBD pour un site web dynamique. Concepts et solutions possibles. Installation de MySQL. Alternatives. Obligations légales et déclaration CNIL.

### **Accès à la base**

Automatisation des accès, comptes utilisateurs et privilèges.

### **Requêtes SQL**

Création de tables, insertion et recherche de données, mise à jour et suppression. Sécurisation des requêtes SQL.

## **Conclusion**

Discussions et expérimentations libres sur l'ensemble des thèmes abordés.

## C embarqué pour micro-contrôleur

Référence Formation : « PCE »

Filière de formation : « Langages »

*Le langage C est largement employé dans de nombreux développement logiciels, tout particulièrement dans le domaine industriel.*

*Permettant un développement à un niveau élevé, tout en permettant un contrôle très fin du code produit, le langage C est un outil de choix pour le développement embarqué, tant pour des applications s'appuyant sur des systèmes d'exploitation complets (VxWorks, Linux, ucLinux, etc.) que pour les programmes embarqués dans des micro-contrôleurs.*

*L'objectif de ce cours est de maîtriser l'environnement de développement et le langage C pour employer au mieux les fonctionnalités offertes par les micro-contrôleurs modernes.*

### Organisation

---

**Durée** : 4 jours (28 heures).

**Pré-requis** : le stagiaire doit avoir des notions générales de programmation.

**Conseil cursus** : en préambule à cette formation, nous vous conseillons notre formation « Introduction à la Programmation ».

### Thèmes abordés

---

- **Concepts du langage C** : présentation, avantages et inconvénients, utilisations typiques.
- **Outils de développement** : environnement, compilateur, débogueur.
- **Structures des programmes** : exemple, vocabulaire, fonctions et variables, constantes, types et expressions.
- **Les variables du C** : globales et locales, détail des types scalaires, tableaux.
- **Les fonctions** : passage d'arguments, types de retour, passage par valeur, passage par référence.
- **Les pointeurs** : déclarations, pièges, arithmétique des pointeurs, initialisation et utilisation.
- **Structures de contrôle** : tests et itérations, sélections.
- **Expressions du C** : opérateurs mathématiques, logiques et binaires.

### Travaux pratiques

---

Ce cours s'appuie sur de très nombreux exemples et travaux pratiques. Les exercices sont réalisés sur plateforme PIC ou ATMEL au choix des participants.

Les travaux pratiques proposés couvrent l'ensemble des possibilités des micro-contrôleurs (entrées - sorties, conversions analogiques / numériques, communications séries, etc.)

*Plan détaillé au verso ►*

## Plan détaillé

---

### Concepts du langage C

#### Présentation

Premières versions du langage C, évolutions, standards et normes

#### Avantages et inconvénients

Efficacité et performance, proximité du matériel et du processeur, portabilité, laxisme du langage, exemples de bogues courants et des conséquences désastreuses.

#### Utilisations typiques

Systèmes industriels et embarqués, serveurs et réseaux, télécommunication, gestion de données.

### Outils de développement

#### Environnements

Présentation des environnements de développement ATMEL (Starter Kit STK 500) et PIC (PICDEM 2 PLUS). Présentation du compilateur C.

#### Compilation

Création d'un projet et compilation d'un premier programme, exécution et résultat. Rôle de la bibliothèque C.

#### Débogage

Principes de débogage, ICE et traces, tests.

### Structures des programmes

#### Aspect et vocabulaire du C

Présentation de code, indentation, caractères spéciaux, mots-clés, définition et déclaration de fonctions, fichier d'entête, fonctions de bibliothèque.

#### Fonctions

Écriture des fonctions, différences entre déclaration et définition, paramètres.

#### Variables

Types des variables, déclaration et utilisation, expression et constantes.

### Les variables du C

#### Portée et persistance

Stockage des variables en pile, variables globales et variables locales, variables automatiques et statiques.

#### Les types scalaires

Format et représentation interne, utilisation.

#### Les tableaux

Tableaux d'entiers, déclaration, initialisation et utilisation.

### Les fonctions

#### Passage d'arguments

Utilisation de la pile, types des paramètres, modification des paramètres formels, passage par valeur, réservation de l'espace dans la pile.

#### Types de retour

Utilisation des valeurs de retour des fonctions. Limitation à une valeur.

#### Passage d'argument par référence

Nécessité de passer un pointeur dans la pile, modification des paramètres d'appel.

### Les pointeurs

#### Déclarations des pointeurs

Exemple de déclaration des types, affichage des valeurs des pointeurs et des contenus pointés. Risques liés à l'utilisation des pointeurs.

#### Arithmétique des pointeurs

Concepts, vérification des pas d'incrément en fonction des types de données.

## Structures de contrôle

### Structures de test

Structure if / else. Imbrications et indentation. Expression testée. Confusion entre égalité et affectation.

### Structure de sélection

Construction switch / case. Utilisation du break.

### Itération conditionnelle

Boucles while et until. Boucle infinie. Rupture de séquence.

### Enumération

Boucle for. Syntaxe usuelle et variantes. Rupture de séquence.

## Expressions du C

### Opérateurs arithmétique

Opérateurs classiques, notations condensées, pré- ou post- incrémentation et décrémentation.

### Opérateurs logiques

Valeurs de vérité, composition d'expressions booléennes

### Opérateurs binaires

ET, OU, OU exclusif, négation, décalage, masques

## Conclusion

Discussions et expérimentations libres sur l'ensemble des thèmes abordés.

## DEUXIÈME PARTIE

### Thème « Linux »

#### *Filière « Utilisateur Linux »*

- Découvrir et Utiliser Linux
- Utilisateur Linux Avancé
- Écriture de Scripts Shell

#### *Filière « Administrateur Linux »*

- Installer et Administrer une Station Linux
- Administration Linux Avancée
- Administrer un Serveur LAMP

#### *Filière « Linux Industriel »*

- Développement Système sous Linux
- Linux Temps-Réel et Embarqué
- Écriture de Drivers pour Linux
- Portage de Linux sur Plateforme Spécifique (*sur demande*)

La découverte d'un système Linux peut être déroutante pour l'utilisateur n'ayant jamais travaillé dans un environnement Unix. Certaines pratiques courantes, comme l'utilisation des lignes de commandes du shell sont parfois perçues comme contraignantes ou inutilement complexes.

Cette formation brève permet une prise en main rapide et sans difficultés d'une machine Linux par l'utilisateur débutant. En quelques heures ce dernier peut évoluer facilement sur son poste en ayant assimilé toutes les notions essentielles concernant le fonctionnement d'un système Unix.

### Organisation

---

**Durée** : 2 jours (14 heures)

**Pré-requis** : aucun.

**Conseil cursus** : A la suite de cette formation, on peut aborder sereinement la formation "Utilisateur Linux Avancé ou la formation "Ecriture de Scripts Shell pour Linux.

### Thèmes abordés

---

Ce cours est entièrement interactif : le stagiaire progresse directement en suivant le support de cours et les conseils de l'intervenant. Les thèmes abordés en pratique recouvrent au moins les sujets suivants, mais de nombreuses manipulations supplémentaires sont réalisées en réponse aux questions des stagiaires.

- Connexions locales et distantes, changement d'utilisateur
- Maîtrise des commandes de base, enchaînements (pipelines) et redirections des entrées-sorties
- Exploration de l'arborescence, montage de systèmes de fichiers amovibles, étude des i-noeuds, liens physiques et symboliques
- Manipulation des fichiers, recherche d'informations sur le système et dans des fichiers. Expériences avec les variables du shell et écritures de scripts.

### Plan détaillé

---

#### Notions de base

##### Entités et vocabulaire

Utilisateur, système, *shell*, processus, fichier, démon, etc.

##### Connexions à un système Linux

Connexion en mode graphique, interfaces texte, connexion à distance.

##### Environnements graphiques et terminal texte

Comparaison *KDE* et *Gnome*. *Xterm*, console virtuelle, *telnet* et *ssh*.

##### Utilisateurs et administrateur

Commande *su*, utilisateur *root*

#### Utiliser les commandes Unix/Linux

##### Commandes

Commandes, options et arguments, sensibilité à la casse, essai de plusieurs commandes.

##### Pipelines

Caractère *pipe*, redirections, parallélisme.

.../...

## Comprendre le système de fichiers

### Arborescence standard

Présentation, traits communs Unix/Linux, rôle des répertoires principaux.

### Répertoire de travail

Affichage, *working directory* et *home directory*, déplacement.

### Chemins d'accès

Chemin absolu ou relatif, exemples, caractères spéciaux "." et ".."

### Recherche des commandes

Principe du *PATH*, affichage et modification.

### Montage des systèmes de fichiers

Périphériques amovibles, commandes *mount* et *umount*, importance du démontage

## Manipuler des fichiers

### Création et suppression de fichiers

Plusieurs manières de créer un fichier, options de la commande *rm*.

### Les principaux éditeurs Unix

Editeurs *Vi*, *Emacs*, *Nedit*, *Kwrite*, *Gedit*...

### Déplacement et copie

Création et suppression de répertoire, options de *cp* et *mv*

### Permissions d'accès, propriétaire et groupe

Modes d'accès, significations pour un fichier ou un répertoire, utilisation de *chmod*.

### i-noeud, liens physique et symbolique

Différences entre liens physiques et symboliques, utilisations typiques.

### Rechercher un fichier dans l'arborescence

Options de *find* et actions possibles.

### Rechercher du texte dans un fichier

Options et comportement de *grep*, notion d'expressions régulières, combinaison avec *find*.

### Rechercher des informations sur le système

Utilisation du manuel avec *man*, contenu du manuel Linux, commande *apropos*

## Interagir avec le shell

### Commandes

Options, arguments, redirections

### Enchaînement de commandes

Séquencement, parallélisme, avant-plan et arrière-plan, commentaires

### Syntaxe du shell

Caractères spéciaux et normaux, variables, guillemets et apostrophes.

### Ecriture de scripts

Structure, ligne *shebang*, droits d'exécution, exemples.

## Commandes Unix

Liste commentée des principales commandes Unix, avec leurs options utiles au quotidien

### Fichiers et répertoires

*cd*, *pwd*, *ls*, *mkdir*, *rmdir*, *rm*, *cp*, *mv*, *ln*, *touch*, *chown*, *chgroup*, *chmod*, *find*.

### Contenu des fichiers

*cat*, *head*, *tail*, *less*, *more*, *grep*, *xargs*.

### Commandes diverses

*man*, *echo*...

### Commandes d'administration

*mount*, *umount*, *df*, *free*, *ps*, *top*, *kill*.

## Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

### Travaux pratiques

Expérimentations libres suivant les demandes des participants.

La richesse de l'environnement Linux et la quantité phénoménale d'applications disponibles ont un revers : il est souvent difficile de connaître l'ensemble des commandes et des possibilités du système.

Ce cours est conçu pour un utilisateur Linux déjà familiarisé avec les manipulations élémentaires (connexion, manipulation des fichiers, déplacements dans l'arborescence, exécution de commandes simples...). Il pourra ici découvrir les diverses solutions qui s'offrent à lui en réponse aux problèmes quotidiens, avec leurs avantages et leurs inconvénients.

L'approche originale de cette formation consiste à présenter simultanément les applications graphiques les plus récentes et les utilitaires classiques en mode texte que l'on peut automatiser à l'aide de scripts shell. Ceci donne au participant des connaissances et des compétences polyvalentes.

### Organisation

---

**Durée** : 3 jours (20 heures).

**Pré-requis** : connaissance basique des commandes Linux.

**Conseil cursus** : A la suite de cette formation, on peut compléter ses connaissances avec la formation "Ecriture de Scripts Shell pour Linux, ou aborder la filière des formations à l'Administration Linux.

### Thèmes abordés

---

- **Environnement de travail** : environnement graphique et console texte, configurer les préférences, trouver de l'aide ;
- **Bureautique** : écrire un texte, dessiner, calculer, créer un fichier PDF ;
- **Multimédia** : affichage de fichiers graphiques, retouche d'images numériques, diffusion de film ou de musique, conversion de formats ;
- **Réseau** : informations techniques, accès à Internet, connexion à distance, partage entre machines ;
- **Édition de fichiers** : contenu, éditeurs de texte, jeux de caractères, outils *Grep* et *Sed* ;
- **Système de fichiers** : actions récursives et globales sur des répertoires, compression, archivage, création de CD, DVD et accès aux clés USB ;
- **Partage de données** : gestion de versions, fichiers *patches*, groupes d'utilisateurs ;
- **Fonctionnement du système** : *boot* et *shutdown*, services, utilisateurs, ressources ;
- **Tâches** : processus actifs, priorités, signaux, automatisation de tâches.

### Travaux pratiques

---

Les travaux pratiques accompagnant ce cours sont très nombreux et la gamme d'exercices corrigés proposés permettra à chacun de progresser à son rythme en fonction de ses connaissances préliminaires.

## Plan détaillé

---

### Introduction

Présentation rapide de Linux et des autres Unix. Validation des connaissances préliminaires et détermination des buts spécifiques des participants.

### Connaître son environnement de travail

#### Travailler dans un environnement graphique :

Systèmes *Kde*, *Gnome*, *XFCE*, *Compiz*.

#### Travailler sur une console texte

Consoles virtuelles, fenêtres *X-Term*, *Konsole*, *Gnome Terminal*, émulateurs *VT100*, sessions *ssh*. Shell : commandes, options, arguments, redirections, *pipeline*.

#### Configurer les préférences

Préférences, menus, *window managers*. Fichiers de configuration des applications graphiques et texte.

#### Trouver de l'aide

Aide en ligne, manuel Linux, documents de référence, sites web principaux.

### Réaliser des tâches bureautiques

#### Ecrire un texte

Aperçu des traitements de texte des suites *OpenOffice.org*, *Gnome Office*, *Koffice*. Principe du processeur de texte *LateX*.

#### Dessiner

Dessins vectoriel et *bitmap*. Tracé de courbes mathématiques.

#### Calculer

Tableaux des suites bureautiques. Calculettes. Arithmétique du shell, utilitaire *Bc*.

#### Créer un fichier au format PDF

Exportation depuis *OpenOffice.org*, ou conversion avec les utilitaires *Ghostscript*, *Tiff2pdf*, *Pdflatex*...

### Manipuler des fichiers multimédias

#### Afficher des fichiers graphiques

Présentation de *Xv*, *Gthumb*, *Kphotoalbum*, *Qiv*...

#### Importer et retoucher des images numériques

Depuis un scanner ou un appareil photo numérique. Convertir un fichier *raw*. Retouches graphiques et traitements automatisés.

#### Voir un film et écouter de la musique

Lire un DVD ou un fichier *DivX*. Lecture de fichiers *MP3*. Conversions et réglages sonores.

#### Convertir des formats graphiques

Avec *Gimp* en mode graphique, ou *Netpbm* en console.

### Utiliser le réseau

#### Obtenir les informations techniques sur réseau

Adresses IP, table de routage, serveur DNS. Vérification des informations.

#### Accéder à Internet

Naviguer sur le Web. Envoyer un courrier électronique. Se connecter sur une messagerie instantanée.

#### Se connecter à distance

Protocoles disponibles, notions de sécurité.

#### Partager des données entre machines

Transférer des fichiers. Synchroniser des répertoires. Accéder à des répertoires sur d'autres machines Linux ou Windows.

.../...

## Editer des fichiers

### Déterminer le contenu d'un fichier

Recherche du type de fichier, examen de son contenu.

### Editer un fichier texte

Editeurs en mode texte : *Vi*, *Emacs*, *Joe*, ou en mode graphique, *Nedit*, *Kwrite*, *Gedit*, *Xemacs*, *Gvim*...

### Convertir les jeux de caractères

Jeux UTF-8, Latin-1 et Windows.

### Utiliser Grep et Sed

Chercher des chaînes de caractères, remplacer ou supprimer des portions de chaînes.

## Utiliser le système de fichiers

### Modifier récursivement les droits et les appartenances

Composition de *find* et *chmod*, *chown*, *chgrp*.

### Renommer tous les fichiers d'un répertoire

Structure *for/do/done* du shell et opérateur *\$*. Conversion minuscules et majuscules.

### (Dé)compresser des fichiers ou des répertoires

Comparaison des performances des outils de compression standard. Création d'archive *tar* et *cpio*.

### Copier ou graver un CD ou un DVD

En environnement Gnome ou Kde. Utilitaires en ligne de commande pour automatiser.

### Accéder au contenu d'une clé USB

Recherche du nom de périphérique, montage et démontage.

## Partager des informations

### Utiliser les outils de gestion de versions

Connaître Rcs et Cvs.

### Manipuler les fichiers de différences (*patches*)

Créer un fichier de différences. Appliquer un *patch*.

### Comprendre les groupes d'utilisateurs

Création de groupes et modifications d'appartenance des utilisateurs et des fichiers.

## Comprendre le fonctionnement du système

### Démarrer et arrêter Linux

Notion de *boot* et de *shutdown*. Les démons et les services réseau.

### Découvrir les autres utilisateurs

Autres utilisateurs connectés, non-connectés.

### Voir les ressources disponibles

Taille mémoire, espace disque, quotas et limites par utilisateur.

## Gérer mes tâches

### Voir les processus actifs

Moniteur système, commandes, répertoire */proc*. Avant-plan et arrière-plan sur une console texte. Priorités.

### Geler temporairement un processus

Depuis un terminal, ou avec le moniteur système, envoi de signal. Reprise du processus gelé.

### Tuer définitivement un processus

Depuis un terminal, depuis le moniteur système ou depuis une console texte.

### Automatiser des tâches

Utilisation de la *crontab*.

Le shell représente à la fois l'interface frontale du système Unix et un véritable langage de programmation. Sa maîtrise est un atout essentiel pour l'utilisateur et l'administrateur Unix.

Ce cours met en relief les aspects importants de l'écriture d'un script à la fois robuste et performant, mais également portable sur les différents systèmes Unix actuels. On trouvera en outre dans ce cours une brève introduction aux langages de scripts Sed et Awk, qui sont très utilisés pour étendre les possibilités des scripts shell.

## Organisation

---

**Durée :** 3 jours (20 heures)

**Pré-requis :** Utilisateur habitué au système Linux.

**Conseil cursus :** A la suite de cette formation, on peut compléter ses connaissances avec la formation "Utilisateur Linux Avancé", ou aborder la filière des formations à l'Administration Linux.

## Thèmes abordés

---

- **Programmation par scripts :** les différents shells, les contraintes liées à l'écriture de scripts ;
- **Fonctionnement du shell :** analyse détaillée de l'interprétation des lignes de commande par le shell ;
- **Déroulement des scripts :** enchaînement des commandes, structures de contrôle ;
- **Commandes Unix standards :** survol des commandes essentielles et présentations des commandes avancées utiles ;
- **Bonne écriture d'un script :** règles pratiques pour améliorer la qualité, la robustesse et la lisibilité des scripts shell ;
- **Expressions rationnelles :** mise en pratique avec l'utilitaire Grep des expressions rationnelles (expressions régulières) augmentant la portée des scripts shell ;
- **Langages Sed et Awk :** présentation rapide des commandes les plus utiles des langages Sed et Awk.

## Plan détaillé

---

### Première journée

#### 1ère séquence 9h00-10h30

**Introduction :** principes des shells Unix, shells Bourne et C, standard SUSv3.

#### 2ème séquence 10h45-12h15

**Exécution des scripts :** interprétation et compilation, éditeurs de texte, invocation des scripts, permissions d'exécution, fonctionnement de la ligne shebang.

#### Travaux pratiques

Interpréteurs de commandes disponibles, éditeurs de textes, invocation d'un script, chemin de recherche, variable PATH et cheval de Troie, shell interactif et shell de script. Utilité de la ligne shebang.

#### 3ème séquence 13h15-15h15

**Interprétation d'une commande :** boucle générale, lecture analyse, exécution.

**Étapes d'analyse :** accolades, tilde, remplacement des paramètres, substitution des commandes, arithmétique, développement des noms de fichiers.

#### Travaux pratiques

Manipulation des variables, protection des expressions, caractères

génériques, invocation de commandes et évaluation arithmétique.

#### 4ème séquence 15h30-17h30

**Redirections des entrées-sorties** : entrées-sorties standards d'un processus, redirections, tube, document en ligne.

**Variables et environnement** : variables, tableaux, paramètres, environnement d'un processus.

##### Travaux pratiques

Redirection des sorties, paramètres de la ligne de commande, extraction de préfixes et suffixes.

## Deuxième journée

#### 1ère séquence 9h00-10h30

**Commandes** : commandes simples, pipelines, listes de pipelines, commandes composées, fonctions, sous-shell.

#### 2ème séquence 10h45-12h15

**Structures de contrôle** : sélection et tests, itération de liste et boucles.

##### Travaux pratiques

Structures de boucles for et while, comptage à rebours, parallélisme, priorités, boucles imbriquées, tests des caractéristiques d'un fichier, mise en correspondance de chaînes, fonctions récursives.

#### 3ème séquence 13h15-15h15

**Exécution des commandes** : ordre de recherche, exécution de commandes binaires et de scripts dans un processus fils ou dans le processus père, commandes essentielles pour les scripts (entrées-sorties, interactions avec le système, configuration du shell, arguments de ligne de commande).

**Commandes Unix standards** : manipulation des fichiers, contenu des fichiers, interactions avec le système.

##### Travaux pratiques

Saisie renforcée d'une réponse de l'utilisateur, choix d'une option, conversion entre minuscules et majuscules, lecture d'un fichier.

#### 4ème séquence 15h30-17h30

**Bonne écriture d'un script** : présentation générale, commentaires, variables, indentation, utilisation des fonctions.

**Améliorations supplémentaires** : gestion des erreurs, messages de débogage, bibliothèques de fonction.

##### Travaux pratiques

Automatisation de transfert FTP, études des processus créés par les commandes Unix, arithmétique avancée.

## Troisième journée

#### 1ère séquence 9h00-10h30

**Utilitaire Grep** : fonctionnement de Grep, variantes et options, principe des expressions rationnelles (expressions régulières), association entre Find et Grep.

**Expressions rationnelles simples** : caractères normaux et spéciaux, listes et intervalles, classes de caractères.

#### 2ème séquence 10h45-12h15

**Expressions rationnelles étendues** : répétitions, alternatives et groupements, références arrières.

**Récapitulatif sur les expressions régulières**

##### Travaux pratiques

Options de Grep, correspondance d'expressions rationnelles, écriture d'expression simple, construction progressive d'expressions rationnelles complexes.

#### 3ème séquence 13h15-15h15

**Langage Sed** : introduction, commandes essentielles, commandes supplémentaires, commandes complexes.

#### 4ème séquence 15h30-16h30

**Introduction à Awk** : présentation, essentiel de Awk, structures de contrôle, opérateurs, fonctions, exemple complet.

##### Travaux pratiques

Remplacement de chaîne dans plusieurs fichiers, extraction de portions de fichier, extraction d'une ligne donnée, remplacements de motifs complexes, élimination de balises HTML

# Installer et Administrer une Station Linux

Référence formation : ALI

Filière : « Linux – Administration »

*L'installation de Linux sur un poste de travail à partir d'une distribution moderne est une opération relativement simple, mais il arrive parfois que certains concepts semblent obscurs (swap, bootloader, partitionnement, etc.)*

*La première journée de cette formation est consacrée à l'installation de Linux (à partir de distributions libres, au choix : Fedora, Ubuntu, Mandriva, Debian...) et à la bonne configuration du poste de travail.*

*Les deux jours suivants sont consacrés à l'optimisation du système, à l'installation d'applications supplémentaires et à la mise en oeuvre de différents services (firewall, serveur FTP, serveur HTTP, client et serveur Samba, etc.) qui transformeront notre station individuelle isolée en un véritable poste de travail complet, pouvant garantir une sécurité et une efficacité optimales.*

## Organisation

---

**Durée** : 3 jours (20 heures).

**Pré-requis** : connaissance basique des commandes Linux.

**Conseil cursus** : A la suite de ce cours, il est possible de compléter ses compétences avec la formation « Administration Linux Avancée » ou de demander une préparation à la certification LPI 101.

## Thèmes abordés

---

- **Concepts** : Linux, logiciels libres, distributions, préparation d'une installation ;
- **Installation basique** : partitionnement, formatage, *bootloader*, *packages*, utilisateurs, réseau ;
- **Finalisation d'installation** : réseau, mise à jour, ajout de *packages* ;
- **Outils de développement** : utilisation de la chaîne de compilation pour installer des *packages* fournis sous forme source ;
- **Configuration réseau** : configuration des adresses, du routage, du serveur DNS, des passerelles, du firewall ;
- **Serveur réseau** : installation des serveurs FTP et HTTP ;
- **Cohabitation avec Windows** : installation et configuration de Samba.

## Travaux pratiques

---

*Les travaux pratiques accompagnant ce cours sont très nombreux et la gamme d'exercices corrigés proposés permettra à chacun de progresser à son rythme en fonction de ses connaissances préliminaires.*

*Plan détaillé au dos ►*

## Plan détaillé

---

### Introduction

#### Linux et les logiciels libres

Présentation rapide de Linux et des autres Unix. Validation des connaissances préliminaires et détermination des buts spécifiques des participants.

#### Les distributions

Présentation des différentes distributions majeures, choix d'une distribution et préparation des CD d'installation.

#### Travaux pratiques

Vérification des versions disponibles, gravure des CD à partir de dépôts locaux.

### Installer Linux

#### Préparation de l'installation

Choix du partitionnement, cohabitation avec une partition Windows, *dual-boot*.

#### Démarrage de l'installation

Partitions, formatage, informations basiques (fuseau horaire), utilisateurs et administrateur, configuration réseau.

#### Installation des packages

Choix du type d'installation, survol des packages disponibles et sélection, installation des packages.

#### Test de l'installation

Vérification de la réussite de l'installation, *checklist* de validation.

#### Travaux pratiques

Avec ce chapitre, le stagiaire réalise une installation de Linux sur un poste de travail contenant une partition Windows qu'il faudra conserver.

### Affiner son installation

#### Mise à jour

Vérification et mise à niveau du système.

#### Environnement de travail

Choix de l'environnement, sélection des applications nécessaires.

#### Packages

Manipulation des gestionnaires de *packages* de la distribution.

#### Utilisateurs

Ajout d'utilisateurs et de groupes. Fonctionnement des mots-de-passe.

#### Travaux pratiques

Choix d'un environnement graphique. Recherche et installation de *package* manquant. Application des correctifs de sécurité, mise à niveau du système.

### Utiliser les outils de développement

#### Outils

Principe de la chaîne de compilation *GNU*, utilitaires fournis.

#### Utilisation

Compilation de *packages* sous forme source.

#### Exemples

Modification et personnalisation de *packages* source

#### Travaux pratiques

Utilisation de la chaîne de compilation pour installer un *package* disponible uniquement sous forme de code source.

.../...

## Comprendre la configuration réseau

### Adresses réseaux

Fichiers et services de détermination d'adresse, client *DHCP*.

### Routage et DNS

Passerelle, configuration du *DNS*.

### Firewall

Configuration du *firewall*, règles de protection.

### Travaux pratiques

Modification des adresses d'une machine. Visualisation des effets de la configuration du *firewall*.

## Installer des serveurs

### Serveur HTTP

Installation et test du serveur *Apache*.

### Serveur FTP

Installation et test du serveur *VSFTP*.

### Serveur SSH

Principe, installation, sécurité des clés, utilitaires associés

### Travaux pratiques

Installation d'un serveur *Apache* avec des pages *web* simples. Installation d'un serveur *FTP* anonyme. Utilisation de *SSH* pour des connexions distantes.

## Cohabiter avec Windows

### Dual Boot

Configuration détaillée des bootloaders *Grub* et *Lilo*. Accès aux partitions Windows *VFat* et *NTFS* depuis Linux.

### Serveur Windows

Accès depuis Linux aux répertoires partagés d'un serveur Windows

### Serveur Linux

Installation et configuration de *Samba* pour offrir des services aux clients Windows

### Travaux pratiques

Echanges de fichiers entre partitions de la même machine, entre machines différentes par réseau.

## Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

### Travaux pratiques

Expérimentations libres suivant les demandes des participants.

Cette formation de trois jours est prévue pour les utilisateurs déjà habitués aux tâches d'administrations courantes (démarrage de service, mise à jour du système, etc.) et désireux d'acquérir une véritable expertise sur un serveur Linux.

On y traite de l'optimisation, de la configuration réseau, des reprises en cas de crash, des sauvegardes automatisées, et de l'installation de services avancés (annuaires réseau, DNS, serveur mail, etc.).

### Organisation

---

**Durée** : 3 jours (20 heures).

**Pré-requis** : connaissance basique des commandes Linux.

**Conseil cursus** : A la suite de ce cours, il est possible de compléter ses compétences avec la formation « Administration d'un Serveur LAMP » ou de demander une préparation à la certification LPI 201.

### Thèmes abordés

---

- **Introduction** : buts et principes de l'administration d'un système Linux, possibilités et limites ;
- **Optimisation des performances** : audit, analyse de traces, optimisation de la mémoire, des disques, des processus, et compilation du noyau ;
- **Récupération du système** : recherche et correction de pannes, de blocage, de *crashes* intempestifs, etc. ;
- **Protection des données** : journalisation, sauvegardes, LVM et disques Raid ;
- **Performance réseau** : configuration et administration de base, mesure et optimisation, équilibrage de charges, virtualisation ;
- **Services réseau** : LDAP, Samba, SMTP, POP ;

### Travaux pratiques

---

Les travaux pratiques accompagnant ce cours sont très nombreux et la gamme d'exercices corrigés proposés permettra à chacun de progresser à son rythme en fonction de ses connaissances préliminaires.

Plan détaillé au verso ►

## Plan détaillé

---

### Introduction

#### Concepts

Buts et principes de l'administration d'un système Linux. Détermination des objectifs et des connaissances préliminaires des participants.

#### Possibilités et limites

Notion de qualité de service

### Optimisation des performances

#### Audit

Performances matérielles. Charge du système. Surveillance des processus.

#### Analyse de traces

Fonctionnement des démons et des traces du système.

#### Optimisation de la mémoire

Organisation de la mémoire. Partition ou fichier de *swap*. Paramètres de réglage du noyau.

#### Optimisation des disques

Mesure de performance. Choix d'un ordonnanceur des entrées-sorties. Réglage du DMA. Paramètres du système de fichiers.

#### Optimisation du multitâche

Priorités des processus. Temps-partagé et temps-réel. Groupes de CPU et répartition des tâches.

#### Compilation du noyau

Téléchargement des sources. Configuration et optimisation. Compilation et installation du nouveau noyau.

### Récupération du système

#### Recherche et correction de pannes

Diagnostic d'un système refusant de démarrer. Bios, *Bootloader*, processus *init*, modules du noyau, partition racine.

#### Débogage en cas de blocage

Saturation de la mémoire ou de la charge processeur. Mise en service du *watchdog* intégré. Utilisation des touches magiques *Sysreq*.

#### Crashes intempestifs

Utilisation d'une console distante. Traces déportées.

### Protection des données

#### Journalisation

Options des systèmes de fichiers ext3, reiserfs, vérifications du disque, sauvegarde du journal.

#### Sauvegardes

Politique de sauvegarde des données. Synchronisation. Outils de sauvegarde. Amanda. Sauvegarde, vérification et restauration.

#### LVM

Principes. Création et suppression de volumes logiques. Utilisation des groupes de volumes. Redimensionnement.

#### Disques Raid

Raid matériel et logiciel. Installation d'un disque Raid 0 et Raid 5. Remplacement de disque Raid défaillant.

.../...

## Performances réseau

### Configuration et administration de base

Outils d'administration.

### Mesure de performances et optimisation

Surveillance du trafic, organisation du réseau. Équilibrage de charge matériel et logiciel.

### Virtualisation

Xen et KVM

## Services réseau

### Annuaire LDAP

Principe. Installation. Tests de fonctionnement. Paramétrage.

### Serveur Samba

Utilisations de Samba. Installation et test. Partage de fichiers. Partage d'imprimante.

### Serveur DNS

Aperçu du fonctionnement du DNS. Installation de BIND et configuration de quelques enregistrements.

### Serveur HTTP

Fonctionnement d'un serveur HTTP. Installation d'Apache. Syntaxe des pages HTML. Tests et paramétrage.

### Serveur SMTP

Mécanisme de transmission des courriers électronique. Serveurs Sendmail, Qmail, Postfix. Installation et test. Sécurité d'un serveur SMTP.

### POP

Installation d'un serveur POP. Test de réception de courrier.

## Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

### Travaux pratiques

Expérimentations libres suivant les demandes des participants.

*Le succès du quatuor Linux, Apache, MySQL, PHP est tel que l'acronyme LAMP a été créé pour décrire les serveurs qui l'utilisent.*

*Notre formation de 4 jours est destinée aux stagiaires ayant déjà une formation d'administrateur Linux. On y étudie la configuration fine du serveur Apache, la mise en oeuvre et l'optimisation de la base de données MySQL, et le fonctionnement des scripts PHP pour supporter des sites Web dynamiques performants.*

### Organisation

---

**Durée** : 4 jours (28 heures).

**Pré-requis** : Compétences en administration d'une station Linux.

**Conseil cursus** : A la suite de ce cours, nous vous proposerons d'approfondir vos connaissances avec le cours de Programmation en PHP.

### Thèmes abordés

---

- **Présentation** : introduction, rôles et intérêts des composants d'un serveur LAMP ;
- **Système Linux** : présentation, variantes et distributions, principes d'installation et d'administration, outils disponibles ;
- **Serveur HTTP Apache** : installation d'Apache, test avec des pages HTML statiques, sécurisation du serveur, paramétrage pour une efficacité optimale ;
- **Langage PHP** : nécessité d'utiliser un langage de programmation pour étendre HTML, aperçu de PHP, pages dynamiques, formulaires ;
- **Base de données MySQL** : nécessité d'utiliser une base de données pour la persistance des informations, présentation de MySQL, présentation des alternatives. Aperçu des requêtes SQL. Intégration avec PHP. Tests ;
- **Intégration complète** : construction d'un site web professionnel avec pages de saisie, informations dynamiques, « panier d'achat » et sessions de connexion.

### Travaux pratiques

---

*Les travaux pratiques associés à ce cours sont une construction progressive d'un site web professionnel, que chacun peut développer et étendre en fonction de ses centres d'intérêts personnels.*

*Plan détaillé au verso ►*

## Plan détaillé

---

### Présentation

#### Introduction

Evolution des sites Web. Pages statiques. HTML et CSS. Pages dynamiques.

#### Rôles des composants d'un serveur Web

Système d'exploitation et interface frontale, langage de programmation et stockage des données.

#### Intérêts des outils LAMP

Logiciels et licences libres. Recherche de documentation et d'assistance.

### Système Linux

#### Présentation

Environnement Linux et projet GNU. Applications sous licence GPL.

#### Variantes et distributions

Distributions libres ou commerciale. Comparaisons.

#### Installation et administration

Principe d'installation. Ajout et mise à jour de *packages*.

#### Outils disponibles pour la construction de site Web

Editeurs de texte. Suite bureautique Open Office. Création et traitement d'images avec Gimp.

### Serveur HTTP Apache

#### Installation d'Apache

Installation du *package*, activation du serveur. Fichiers de configuration.

#### Le langage HTML

Aspect d'une page HTML. Balises principales. Feuilles de style CSS. Principe des pages tabulées.

#### Sécurisation du serveur

Risques d'attaque. Vérification des paramètres de sécurité. Mise à jour automatique.

#### Paramétrage

Fichiers de configuration. Optimisation de la charge du serveur. Alternatives.

### Langage PHP

#### Étendre HTML

Limitation des pages statiques. Extensions *client-side* ou *server-side*. Javascript et PHP.

#### Aperçu de PHP

Structure du langage. Cohabitation avec HTML. Syntaxe. Variables.

#### Pages dynamiques

Passage de paramètres. Chargement de portions de page.

#### Formulaires

Formulaires HTML. Validation des saisies. Action en sortie de formulaire.

.../...

## Base de données MySQL

### **Persistance des informations**

Fichiers et bases de données relationnelles. Déclaration des fichiers auprès de la CNIL.

### **MySQL et ses alternatives**

Présentation et évolutions. Oracle, SQLite, PostgreSQL. Installation de MySQL.

### **Requêtes SQL**

Requêtes principales. Création de base, de table, de relation. Recherche dans la base. Mises à jour. Autorisations des utilisateurs.

### **Intégration avec PHP**

Transmission de requêtes SQL avec PHP. Sauvegarde des données. Pages dynamiques.

## Intégration complète

### **Plan d'un site Web professionnel**

Mise en place d'un site Web professionnel simple.

### **Formulaires de saisie**

Création d'un formulaire de demande de renseignement. Enregistrement dans la base de données.

### **Panier d'achat**

Création d'un panier d'achat. Formulaires de validation et de paiement. Envoi d'*email*.

### **Connexions et sessions**

Création de compte utilisateur. Connexion automatique et *cookies*.

## Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

### **Travaux pratiques**

Expérimentations libres suivant les demandes des participants.

*A travers l'étude approfondie des principaux éléments du développement système sous Linux, les participants seront capables d'aborder des projets d'envergure sur l'ensemble des systèmes Unix. Les points essentiels de leur interface de programmation standard (Posix - SUSv3) sont étudiés tant de manière théorique que dans le cadre de travaux pratiques détaillés.*

## Organisation

---

**Durée** : 4 jours (28 heures).

**Pré-requis** : connaissance de Linux (niveau utilisateur) et du langage C.

**Conseil cursus** : pour prolonger cette formation, nous vous proposons la formation « Programmation réseau sous Linux » ou notre filière de formations « Linux Industriel ».

## Thèmes abordés

---

- **Outils et méthodes de développement** : environnements et outils de compilation et mise au point.
- **Déroulement des processus** : création, exécution, attente...
- **Programmation multi-threads** : création et déroulement, synchronisation et communications.
- **Gestion de la mémoire** : mémoire virtuelle, allocation et libération, débogage.
- **Communications entre processus** : mémoire partagée, files de messages, signaux...
- **Aperçu de programmation réseau** : client et serveur TCP/IP.

## Plan détaillé

---

### Outils et méthodes de développement

#### Environnement Linux

Noyaux et distributions, composants d'un système Linux, rôle de la bibliothèque C.

#### Outils de développement

Editeurs et environnement Eclipse, compilateurs et bibliothèques, débogage.

#### Travaux pratiques

Compilations et débogages ; débogage post-mortem ; statistiques d'exécution ; tests de couverture.

### Déroulement des processus

#### Processus Unix

Concepts, identifiants, filiations.

#### Débuts et fins d'un processus

Création, fins normales et anormales, fin d'un processus fils.

#### Exécution d'un nouveau programme

Appels-système exec(), system(), popen(), problèmes de sécurité.

#### Travaux pratiques

Créations de processus ; examen de l'état Zombie ; attente de la fin d'un fils ; exécutions de programmes ; appels de sous-programmes ; exécutions parallèles.

.../...

## Programmer avec les threads Posix.1c

### Threads Posix.1c

Exécution, terminaison et attributs des threads.

### Synchronisation

Mutex, verrous rwlock, conditions.

### Travaux pratiques

Création de threads, synchronisation sur des mutex, partage de données.  
Comparaison des temps de création avec les processus.

## Gestion de la mémoire

### Principes de la mémoire virtuelle

Espace d'adressage et mémoire physique, segmentation, pagination.

### Allocation de la mémoire

Mécanisme d'allocation et de libération, fiabilité.

### Projections en mémoire

Principes, projections de fichiers ou de périphérique.

### Débogage

Configuration des pages, indication d'usage, détection des fuites et débordements mémoire.

### Travaux pratiques

Mémoire virtuelle ; exploration ; variables dynamiques ; projection de fichier ; configuration avec malloc( ) ; utilisation de mmap( ) ; réussite des allocations et saturation de la mémoire ; débordements de chaîne ; protection.

## Communications entre processus

### Mécanismes IPC Posix

Principes, files de messages, segments de mémoire partagée, sémaphores.

### Signaux Unix

Principes, émission, réception, gestionnaire, blocage, signaux temps-réel.

### Tubes de communication

Principes, création et utilisation, duplication, tubes nommés.

### Travaux pratiques

Émission et réception de signaux ; blocage et déblocage ; signaux BSD et Système V ; signaux temps-réel ; files de message ; segments de mémoire partagée ; synchronisation par un sémaphore ; tube simple ; redirection de commandes ; tubes nommés.

## Aperçu de la programmation réseau

### Famille TCP/IP

protocoles en couches, routage, services, résolution de noms.

### Client TCP/IP

socket, connexion, lecture et écriture, fermeture.

### Serveur TCP/IP

multi-processus et multi-threads.

### Travaux pratiques

Résolution d'adresse et services, client et serveurs TCP/IP.

## Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

### Travaux pratiques

Expérimentations libres suivant les demandes des participants.

*L'utilisation de Linux dans les environnements industriels, pour des applications à fortes contraintes temporelles ou sur des systèmes retroints nécessite une bonne connaissance des mécanismes sous-jacents, comme l'ordonnancement des tâches, la gestion mémoire, ou le chargement des drivers du noyau.*

*Ce cours vous propose une exploration en profondeur du système Linux, de ses possibilités - et de ses limites - pour les applications temps-réel et les systèmes embarqués.*

## Organisation

---

**Durée** : 4 jours (28 heures).

**Durée en formation individuelle accélérée** : 3 jours (20 heures).

**Pré-requis** : Connaissance de Linux (utilisateur) et du langage C.

**Conseil cursus** : La formation Ecriture de Drivers pour Linux représente un bon complément à ce cours.

## Thèmes abordés

---

- **Développement industriel sous Linux** : outils de développement et de mise au point des applications, utilisation de l'environnement Eclipse, compilation et débogage croisés pour une cible embarquée, ajustement et compilation du noyau Linux.
- **Linux embarqué** : construction d'un noyau et d'un environnement réduit pour une cible embarquée, étude du boot du système, compilation et exécution des applications personnalisées.
- **Développement multi-tâches sous Linux** : processus et threads, communication et synchronisation entre tâches, gestion de la mémoire.
- **Temps-réel sous Linux** : principe de l'ordonnancement sous Linux, temps-partagé et temps-réel souple sous Linux, solutions temps-réel strictes avec RTAI et Xenomai.

## Plan détaillé

---

### I - Linux en environnement industriel

#### Linux et les logiciels libres

Présentation des concepts, des principes et des pratiques. Projet Gnu. Noyaux et distributions Linux.

#### Licences libres

Principes des GPL, LGPL, BSD... et implications pour le développement industriel.

#### Outils de développement libres

Chaîne de compilation Gnu, outils de débogage et de mise au point.

#### Eclipse et le CDT

Environnement de développement intégré. Création de projet, compilation et débogage.

#### Travaux pratiques

Utilisation du compilateur GCC, effets des différentes options. Détection d'erreurs à la compilation, à l'édition des liens. Débogage en cours de fonctionnement avec GDB. Débogage post-mortem avec GDB. Statistiques d'exécution et tests en couverture. Création et compilation d'application avec Eclipse. Utilisation du débogueur intégré.

.../...

## II - Linux embarqué - Noyau

### Linux sur cible embarquée

Plateformes de développement et d'exécution, type de cibles, utilisation d'un émulateur.

### Chaîne de compilation croisée

Principe, création et mise en oeuvre d'une chaîne de compilation croisée. Utilisation de *Buildroot*. Environnement Eclipse pour la compilation croisée.

### Compilation du noyau Linux.

Choix d'une version. Configuration et compilation.

### Installation sur cible

Transfert de l'image du noyau. Configuration du *bootloader* ou de l'émulateur.

### Système de fichiers

Types de système de fichiers, choix. Formatage et création de l'arborescence. Fichiers spéciaux des périphériques.

### Travaux pratiques

Utilisation d'une chaîne de compilation pour *PowerPC*. Création d'une chaîne de compilation pour processeur *Arm*. Compilation et installation d'un noyau Linux pour cible *Arm*. Préparation d'un système de fichiers minimal.

## III - Linux embarqué - Applications

### Utilitaires système

Fonctionnement du processus *init*. Scripts de démarrage. Compilation de *Busybox*.

### Bibliothèques et éditions des liens

Choix des bibliothèques nécessaires. Compilation de bibliothèques statiques ou dynamiques.

### Débogage et mise au point

Débogage distant avec GDB et Eclipse. *Profiling* et tests en couverture.

### Travaux pratiques

Compilation d'utilitaires système avec *Busybox*. Personnalisation des scripts de démarrage. Débogage et optimisation d'applications. Création de bibliothèques diverses.

## IV - Multi-tâches sous Linux

### Processus et threads

Création, exécution, terminaison, attente d'une autre tâche.

### Communications entre processus

Files de messages Posix et segments de mémoire partagée.

### Synchronisation et notification

Mutex, sémaphores Posix et signaux Unix.

### Éléments temporels

Obtenir l'heure, précision, timers.

### Gestion de la mémoire

Allocation et libération dynamiques, fiabilité et débogage.

### Travaux pratiques

Création de processus et attente de processus fils. Passage à l'état *Zombie*. Création de communication entre threads. Utilisation des IPC Posix (mémoire partagée, sémaphores et files de messages). Test de saturation de la mémoire, désactivation de l'*overcommit*.

## V - Temps-partagé et temps-réel souple sous Linux

### Temps-partagé

Principe, configuration, efficacité, préemptibilité du noyau.

### Temps-réel souple (*Soft Realtime*)

Principe, priorités, ordonnancements RR et FIFO.

### Configuration du temps-réel

Passage en temps-réel, spécificité des noyaux postérieurs au 2.6.21.

### Problèmes temps-réel classiques

Synchronisation des démarrages, inversion de priorité...

### Limites du temps-réel souple

Fluctuation des timers, traitement des interruptions.

### Travaux pratiques

Influences des priorités temps-partagé. Passage en temps-réel. Effets des boucles actives en temps-réel. Mesure de variation des timers.

## VI - Temps-réel strict avec Linux

### Principes du temps-réel strict (*Hard Realtime*)

Noyau standard et extensions *RT-Linux*, *RTAI*, *Xenomai*...

### Installation de *Xenomai* ou *RTAI*

Téléchargement et application des *patches*, compilation et chargement.

### Utilisation de *Xenomai*

Survol des API native et Posix de *Xenomai*

### Travaux pratiques

Compilation et installation de *Xenomai*. Test de précision des timers.

## Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

### Travaux pratiques

Expérimentations libres suivant les demandes des participants.

*Le support pour les périphériques est assuré sous Linux par des pilotes (drivers) dont le code se déroule dans le noyau du système d'exploitation. Il est donc nécessaire pour le développeur amené à écrire ou à tester des pilotes de périphériques de maîtriser les concepts propres à la programmation noyau.*

*Ce cours propose une approche originale, s'appuyant sur l'écriture progressive de pilotes de différents types, pour appréhender les mécanismes parfois complexes (préemptibilité, multiprocesseur, support d'architectures différentes, etc.) inhérents au code exécuté en mode noyau.*

*Outre les périphériques classiques (caractère, bloc, réseau), on étudie certains sous-systèmes du noyau tels que les systèmes de fichiers ou l'ensemble USB.*

### Organisation

---

**Durée** : 4 jours (28 heures).

**Pré-requis** : Connaissance de Linux (utilisateur) et bonne familiarité avec le langage C.

**Conseil cursus** : La formation Linux Temps-Réel et Embarqué représente un bon complément à ce cours.

### Thèmes abordés

---

- **Programmer pour le noyau Linux** : noyau Linux et modules, outils de développement, API et environnement de fonctionnement du noyau ;
- **Périphériques caractère** : principes d'écriture d'un pilote de périphérique, appels-systèmes principaux, accès au matériel, traitement des interruptions, *tasklet*, *workqueue* ;
- **Driver avancé** : attentes, asynchronisme, mémoire, classe de périphériques ;
- **Périphériques en mode bloc** : principes, traitement des requêtes, partitionnement, ordonnancement des entrées-sorties ;
- **Systèmes de fichiers** : principe du *VFS*, inscription d'un nouveau système de fichiers ;
- **Périphériques et protocoles réseau** : interface réseau, enregistrement d'une nouvelle interface, étude de la pile IP ;
- **Périphériques USB** : sous-système USB, enregistrement d'un driver *Interrupt*, études de drivers *Bulk* et *Control*.

► [Plan détaillé au dos...](#)

### Première journée - Programmer pour le noyau Linux

#### 1ère séquence 9h00-10h30

Noyau Linux et modules : Historique du noyau Linux, licence GPL, développement du noyau. Appels-système, modules.

##### Travaux pratiques

Observation des appels-système invoqués par des applications et commandes utilisateur. Manipulation des modules précompilés.

#### 2ème séquence 10h45-12h15

Outils de développement noyau : Organisation des sources, compilation du noyau et des modules. Programmation de modules du noyau, compilation de modules indépendants. Messages du noyau, dépendances entre modules.

##### Travaux pratiques

Compilation et installation d'un noyau 2.6. Application de patches pour débogage noyau. Écriture de modules simples du noyau. Intégration dans la chaîne de compilation du noyau. Passage de paramètres au boot.

#### 3ème séquence 13h30-15h15

Interface de programmation du noyau : Chaînes de caractères, blocs mémoire, fonctions numériques et conversions. Pilotes de périphériques. Éléments temporels et actions différées. Préemptibilité du noyau 2.6.

##### Travaux pratiques

Écriture d'un module d'horodatage. Chronométrer les phases de boot.

#### 4ème séquence 15h30-17h30

Environnement du noyau : Tâches et processus current. Espaces d'adressage. Dialogue avec /proc.

##### Travaux pratiques

Écriture d'un module d'information sur les structures internes des processus.  
Écriture d'un module d'horodatage via /proc.

### Deuxième journée - Écriture d'un driver

#### 1ère séquence 9h00-10h30

Écriture d'un pilote de périphérique : Principe des pilotes de périphérique. Réservation de numéros majeurs et mineurs. Enregistrement du pilote de périphérique. Fonctions de lecture et écriture. Fonctions de paramétrage. Synchronisation des appels-système.

##### Travaux pratiques

Manipulation des fichiers spéciaux. Réservation de numéro majeur. Enregistrement de périphérique. Écriture d'un driver simple. Implémentation des routines de lecture et écriture.

#### 2ème séquence 10h45-12h15

Accès au matériel et interruptions : Accès simple au matériel. Contextes d'exécution : appel-système et interruption. Gestion des interruptions. Différer un traitement en interruption : tasklet & workqueue. Protection des variables globales.

##### Travaux pratiques

Écriture d'un gestionnaire d'interruption.

#### 3ème séquence 13h30-15h15

Fonctions avancées d'un pilote de périphérique : Attentes d'événements. Multiplexage d'entrées-sorties. Gestion de la mémoire.

##### Travaux pratiques

Création d'un périphérique "file de messages" virtuel.

#### 4ème séquence 15h30-17h30

Modèle de périphérique du noyau 2.6 : Création d'une classe de périphérique.  
DMA : Transferts de données par DMA.

## Troisième journée - Périphériques bloc et systèmes de fichiers

### 1ère séquence 9h00-10h30

Périphériques en mode bloc : Principe des périphériques bloc. Ecriture d'un driver. Enregistrement du pilote. Déclaration d'un disque générique. Initialisation de la file de requêtes. Requêtes sur un driver bloc.

### 2ème séquence 10h45-12h15

Driver bloc avancé : Traitement différé. Partitionnement du disque. Sous-système Block du noyau. Ordonnanceur des entrées-sorties

#### Travaux pratiques

Ecriture progressive d'un pilote de disque virtuel, en s'appuyant sur les exemples fournis dans le cours

### 3ème séquence 13h30-15h15

Virtual File System : Organisation du VFS Structures File, Dentry, Inode, et Super-bloc.

### 4ème séquence 15h30-17h30

Nouveau système de fichiers : Enregistrement. Initialisation du super-bloc. Implémentation des appels-système de lecture et écriture. Utilisation du cache en lecture et en écriture. Communication avec le sous-système Block.

#### Travaux pratiques

Ecriture d'un système de fichiers virtuel, simple en analysant les étapes de transfert des données.

## Quatrième journée - Autres types de périphériques

### 1ère séquence 9h00-10h30

Périphériques réseau : Dépendance des interfaces bas-niveau et des protocoles réseau. Périphérique net\_device Enregistrement d'une interface, activation, émission et réception de paquets. Statistiques d'utilisation d'interface

#### Travaux pratiques

Ecriture progressive d'un driver pour périphérique virtuel permettant l'utilisation du protocole IPv4.

### 2ème séquence 10h45-12h15

Protocoles réseau : Principes de la pile IP. Emission et réception de données. Communications avec l'interface bas-niveau.

#### Travaux pratiques

Examen du trajet des données au sein de la pile IPv4 lors de réception et d'émission de données avec le protocole TCP/IP.

### 3ème séquence 13h30-15h15

Périphériques USB : Organisation du sous-système USB de Linux. Implémentation d'un driver de classe Interrupt : Enregistrement d'un driver. Endpoints et types de dialogues. Communication avec les URB. Traitements des écritures successives rapides. Déconnexions intempestives et accès concurrents. Gestion simultanée de plusieurs périphériques.

#### Travaux pratiques

Ecriture d'un driver pour carte d'entrée-sortie Velleman K8055.

### 4ème séquence 15h30-16h30

Autres classes USB : Exemples de drivers pour les modes Bulk, Control et Isochrones.

### Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

#### Travaux pratiques

Expérimentations libres suivant les demandes des participants.