Introduction aux systèmes GNU/Linux

Module IntroLinux – S26E01

Philippe Latu / Université de Toulouse
inetdoc.net

Objectif DevOps

De l’idée à la production.

→ Un flux continu de valeur !

Créer un pipeline fluide et continu de la conception à la mise en production, en mettant l’accent sur la livraison rapide de valeur.

Le plan

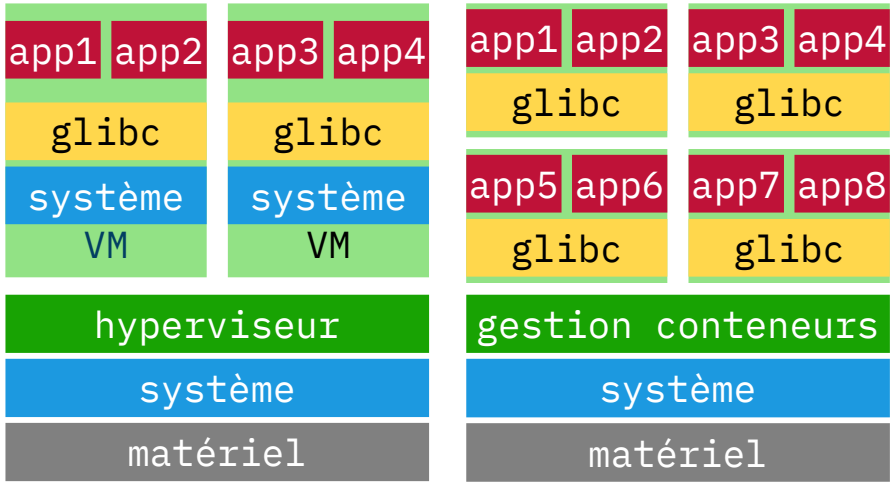
- Progression sur 2 ans
- Concepts Unix et GNU/Linux
- Modélisation d’un système GNU/Linux

Progression sur 2 ans

3e année de licence + 1re année de master

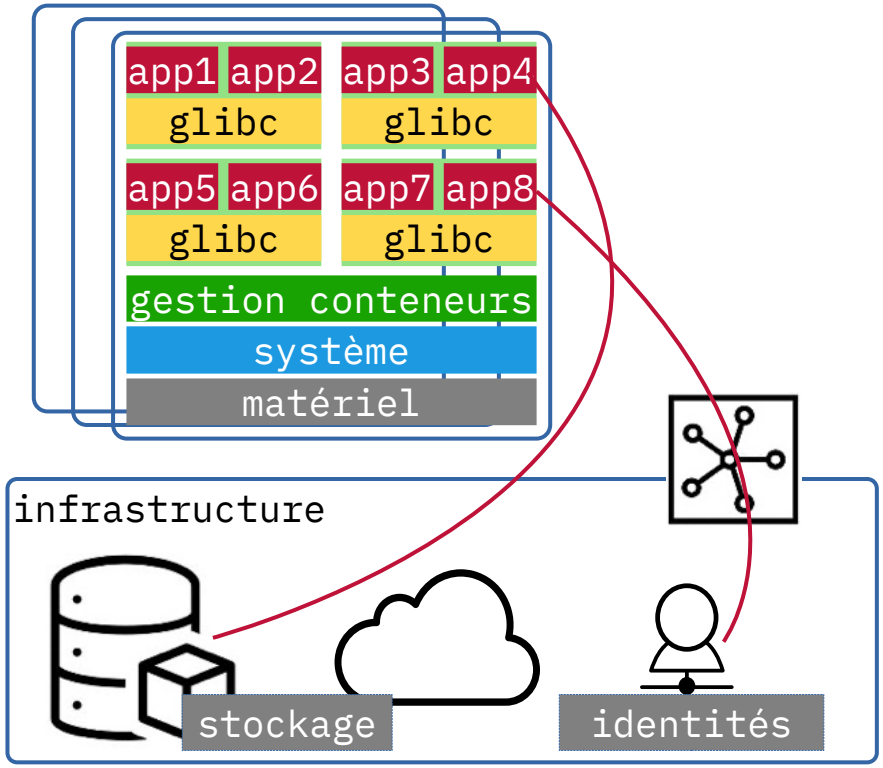
Progression sur 2 ans - L3

- Administrer un système
  - Identifier les composants d'une distribution Linux : Debian
  - Gérer des paquets
  - Gérer des conteneurs système
  - Gérer des machines virtuelles
  - Créer et personnaliser des comptes utilisateurs locaux
  - Identifier les ressources système
  - Gérer les processus



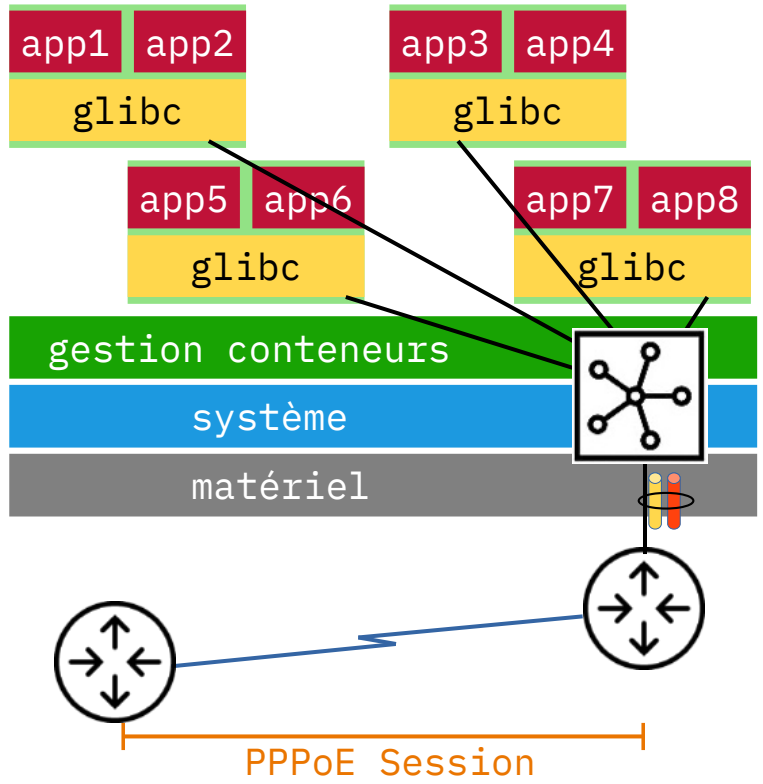
Progression sur 2 ans - M1

- Administrer des systèmes
  - Définir les types de stockage
  - Choisir et configurer une solution de stockage réseau iSCSI ou NFSv4
  - Installer et configurer un annuaire LDAP
  - Associer identité et système de fichiers



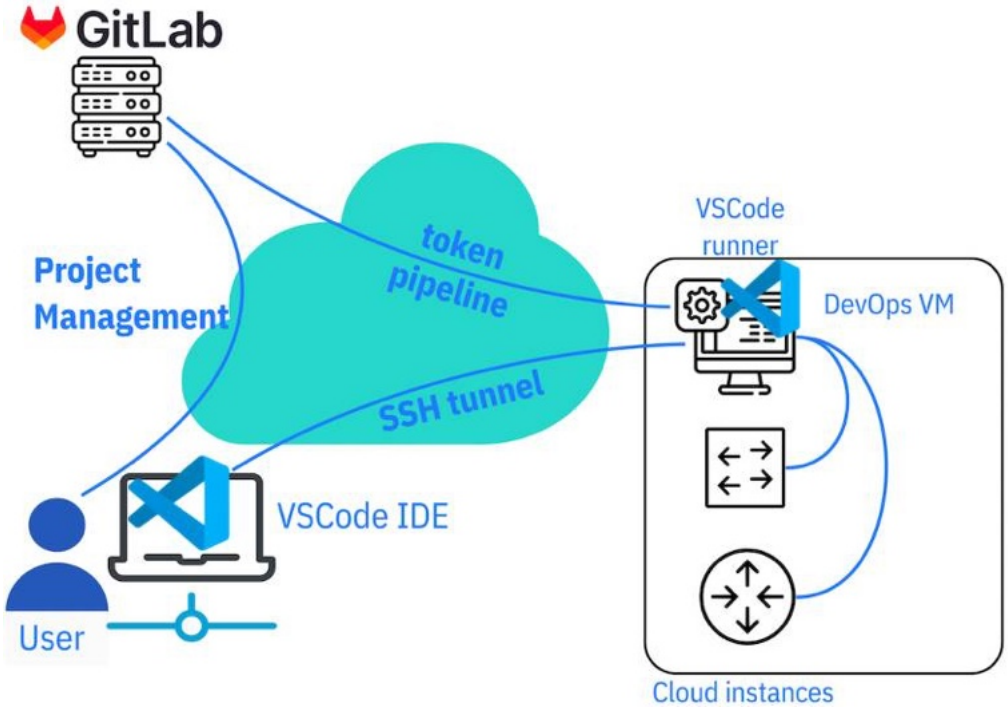
Progression sur 2 ans - M1

- Interconnecter des réseaux hétérogènes
  - Configurer une liaison WAN PPPoE
  - Utiliser le routage inter-VLAN
  - Configurer un réseau de conteneurs
  - Configurer le protocole de routage dynamique OSPF



Progression sur 2 ans - M1

- Automatisation
  - Identifier les modèles de conception et de développement
  - Utiliser des API REST en Python
  - Déployer des conteneurs et des routeurs avec Ansible
  - Construire des pipelines CI/CD avec GitLab

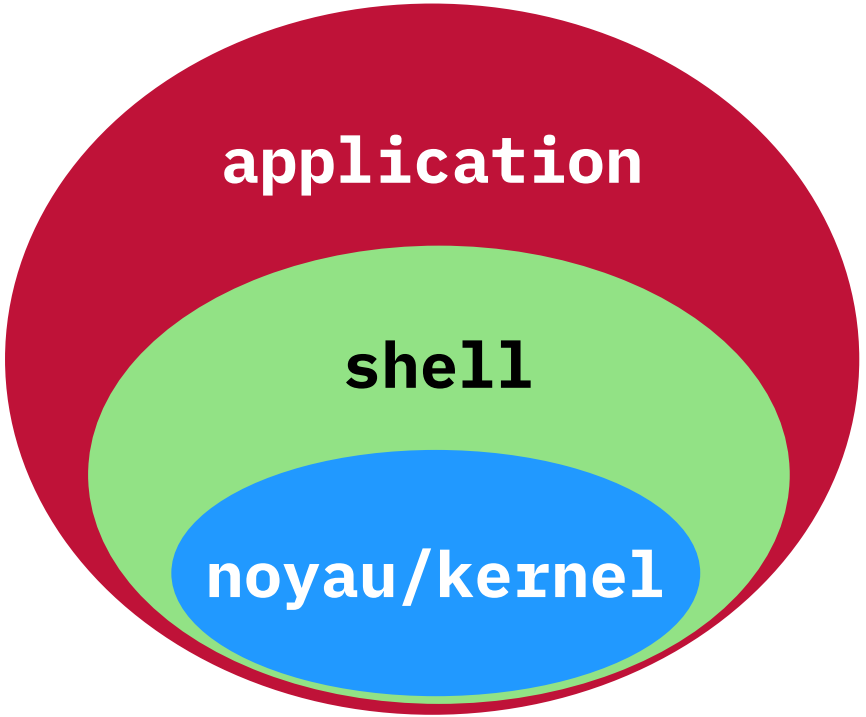


Progression de ce module

3e année de licence

3 séances de cours

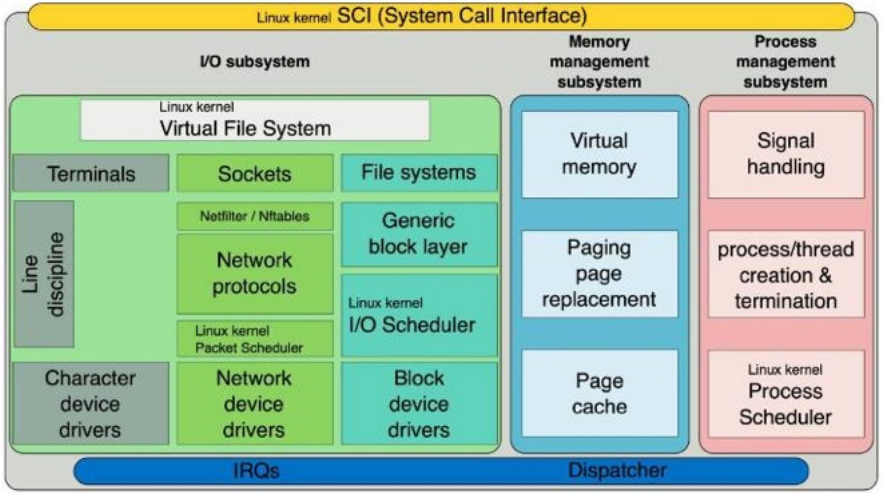
- Cours 1 & 2
  - Définir les 5 fonctions de base des systèmes Unix
  - Définir les 3 couches de la modélisation d'un système d'exploitation
  - Définir les types de virtualisation et de conteneurs
  - Définir une distribution GNU/Linux
  - Identifier les licences



3 séances de cours

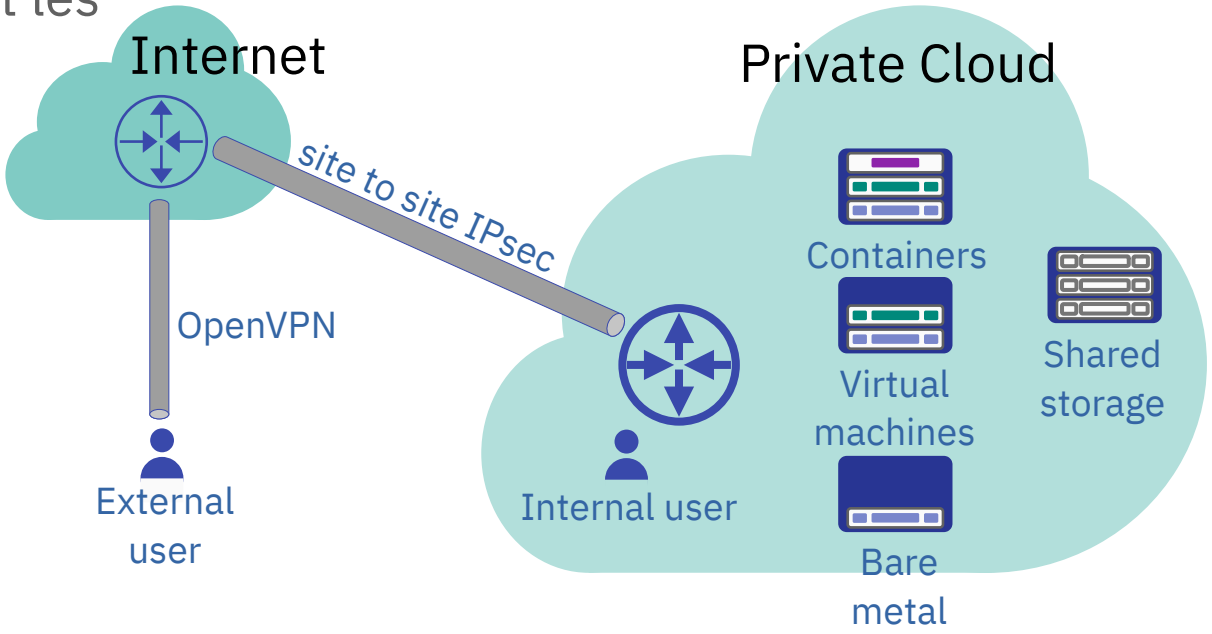
- Cours 3
  - Décrire les protocoles des environnements graphiques
  - Définir les fonctions du shell Bash
  - Définir les permissions sur les fichiers Unix
  - Décrire les outils de l’initialisation système

Initialisation des sous-systèmes du noyau



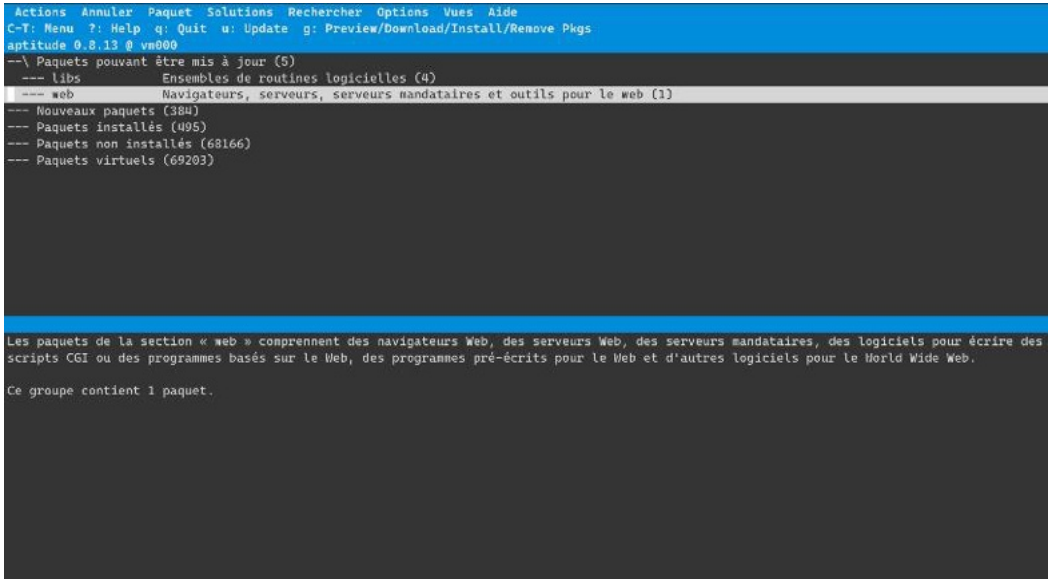
Accéder au cloud privé de la formation

- Travaux Pratiques 1
  - Identifier les types et les modes d'accès aux « Clouds »
  - Utiliser un VPN
  - Lancer une machine virtuelle
  - Configurer un gestionnaire de conteneurs



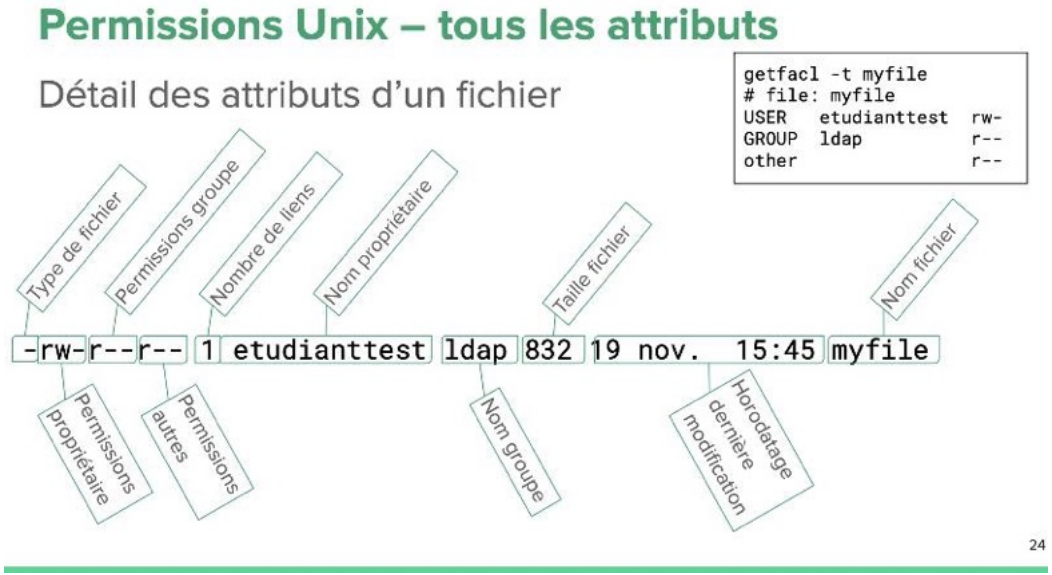
Gérer les paquets et les conteneurs

- Travaux Pratiques 2
  - Définir les fonctions clés de la gestion de paquets
  - Utiliser l'Advanced Package Tool → apt 3.0
  - Utiliser les outils de la gestion de paquets : dpkg, aptitude
  - Installer et configurer le gestionnaire de conteneurs Incus



Gérer les processus et les permissions

- Travaux Pratiques 3
  - Présenter les environnements graphiques
  - Utiliser le shell Bash
  - Gérer les processus
  - Gérer les permissions Unix
  - Définir les processus d'initialisation système



Concepts Unix et GNU/Linux

« Unix est fondamentalement un système d'exploitation simple, mais il faut être un génie pour comprendre sa simplicité. »

Attribuée à Dennis Ritchie

Définir la notion de culture…

« un ensemble lié de manières de penser, de sentir et d'agir plus ou moins formalisées qui, étant apprises et partagées par une pluralité de personnes, servent, d'une manière à la fois objective et symbolique, à constituer ces personnes en une collectivité particulière et distincte. »

Guy Rocher, Introduction à la sociologie générale (1969) – https://fr.wikipedia.org/wiki/Culture

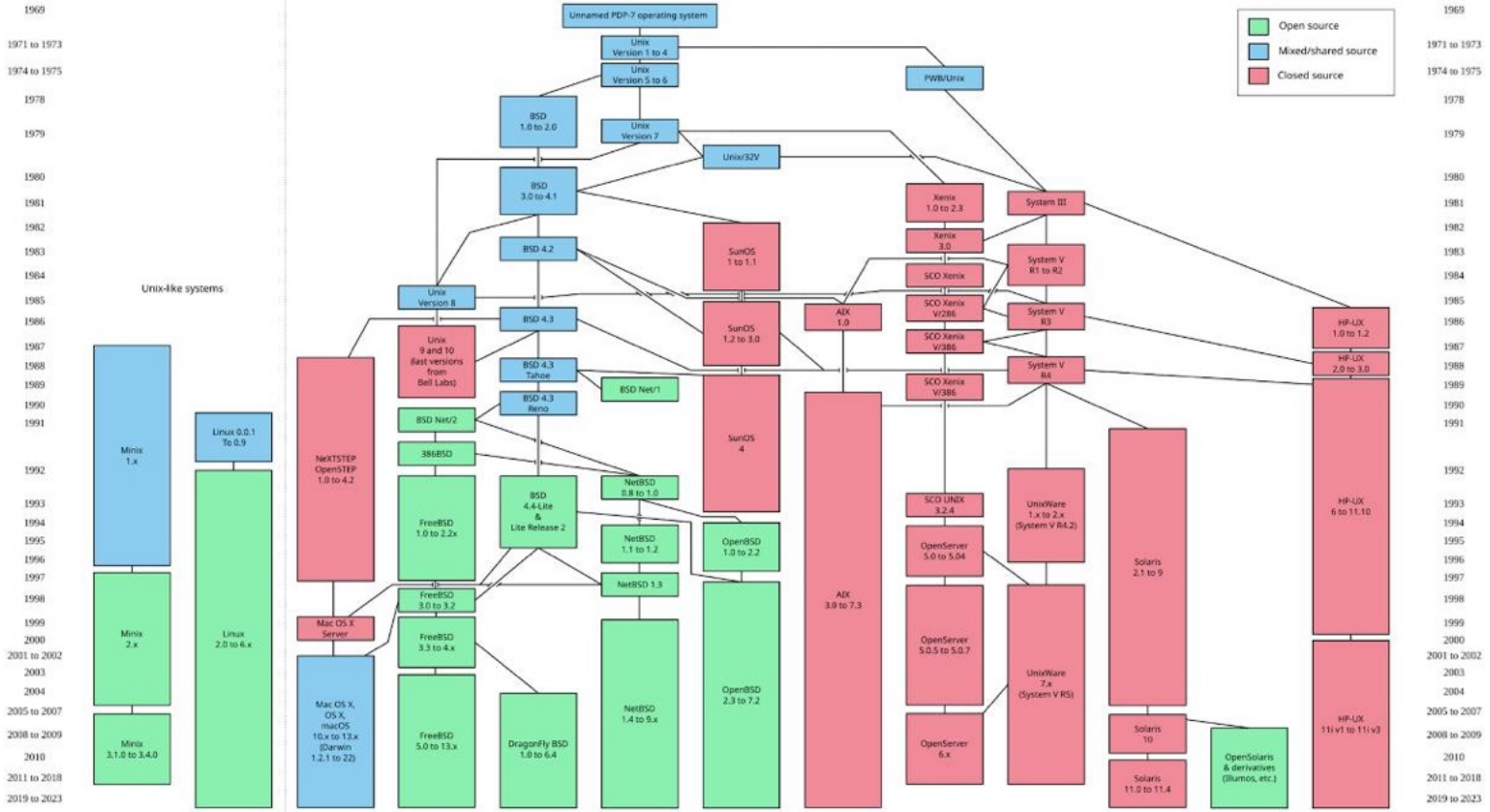
Développer une culture système, c’est…

- Identifier les étapes d’une histoire continue sur près de 6 décennies
- Comparer la genèse des systèmes Unix avec celle des services Internet
- Identifier les processus d’assurance qualité des distributions
- Décrire les nouveaux processus métier → **DevOps !**

Les 5 fonctions de base des systèmes Unix

- Multitâches
  - Temps processeur partagé entre plusieurs programmes
- Multi-utilisateurs
  - Ressources système partagées entre plusieurs utilisateurs
- Portabilité
  - Outils système partagés entre matériels → norme POSIX
- Bibliothèques de développement standard
  - Optimiser la qualité des développements en partageant le code source
- Applications communes
  - Services système, services Internet, etc.

L’histoire Unix



Frise : Wikimedia Commons, Unix_history-simple.svg (CC BY-SA)

1969-1983 – d'Unics à System V

- Unix est un système « accidentel »
  - AT&T Bell Labs – Ken Thompson – Dennis Ritchie
  - 1973 : réécriture en langage C
  - Diffusion sous licence AT&T incluant la totalité du code source
  - 1975 : publication de la RFC 681 NETWORK UNIX
- Apparition des Unix constructeurs
  - Coût matériel + licence prohibitifs
  - Versions constructeurs incompatibles entre elles

1977-1978 – UC Berkeley – BSD

- Branche Unix lancée à partir d'une licence AT&T
  - Nombreuses améliorations
    - Gestion mémoire
    - Sous-système réseau TCP/IP
  - Diffusion entre universités
  - Début de l'Internet universitaire

- Procès USL contre BSDi (1992-1994)
  - BSD → système complet autonome
  - Éclatement de la branche BSD
  - FreeBSD, NetBSD, OpenBSD, DragonFly BSD

1984 – GNU's Not Unix

- Projet lancé par Richard Stallman → 2 objectifs
- Promouvoir les logiciels libres
  - Garantir les libertés des utilisateurs : copyleft, GNU GPL
  - Fédérer les développements libres
  - Applications GNU
- Utiliser les fonctions Unix comme modèle
  - Fonctions de base déjà éprouvées
  - 1990 – chaîne de développement stable
  - GNU C Compiler (GCC)

1991 – Débuts du noyau Linux

- Développement initié par Linus Torvalds
  - « divergences de vues » avec A. S. Tanenbaum
  - https://www.oreilly.com/openbook/opensources/book/appa.html
- Fonctions de base Unix + spécificités
  - Multitâches
  - Multi-utilisateurs
  - Gestion mémoire
    - Paginée et protégée → mémoire virtuelle
    - Isolation entre les processus

2025 – Unix V4 retrouvé

- Une bande magnétique 9 pistes de 1973
  - Trouvée le 28 juillet 2025 à l'Université de l'Utah
  - Lue au Computer History Museum le 19 décembre 2025
- Seule copie complète connue de Unix V4
  - Première version dont le noyau est écrit en langage C
  - Environ 55 000 lignes, dont 25 000 en C
- Démarrable dans l'émulateur PDP-11 SimH
  - https://archive.org/details/utah_unix_v4_raw

2026 – Linux a 35 ans

- 25 août 1991 : annonce de Linus Torvalds
- Linux 7.0 le 12 avril 2026, Linux 7.3 le 18 octobre 2026
  - Une nouvelle version environ tous les 2 mois
  - Version LTS 6.18 maintenue jusqu’en décembre 2028
- Rust n’est plus expérimental
  - Décision du Maintainers Summit de décembre 2025
  - Deuxième langage du noyau, à côté du C
- Debian 13 « trixie » : noyau 6.12 LTS
  - Testing « forky » suit déjà la branche 7.x

Modélisation d’un système GNU/Linux

Le modèle en couches

Noyau/kernel → 3 fonctions

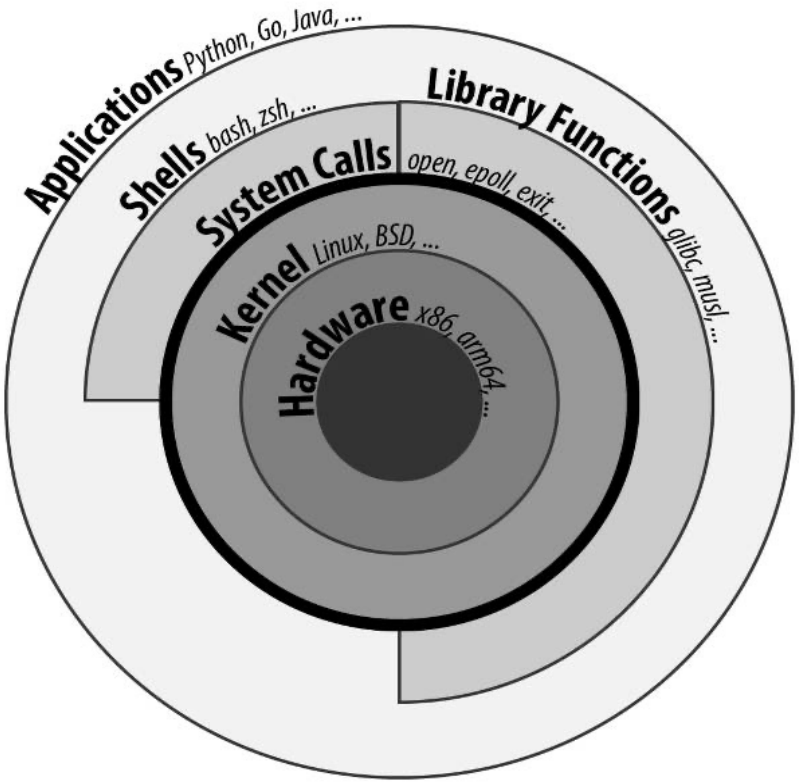
- Temps processeur
- Mémoire
- Entrées/Sorties

Shell → 2 fonctions

- Séquencement entre processus
- Ligne de commande

Application

- Tous les processus utilisateur



Le modèle simple en 3 couches

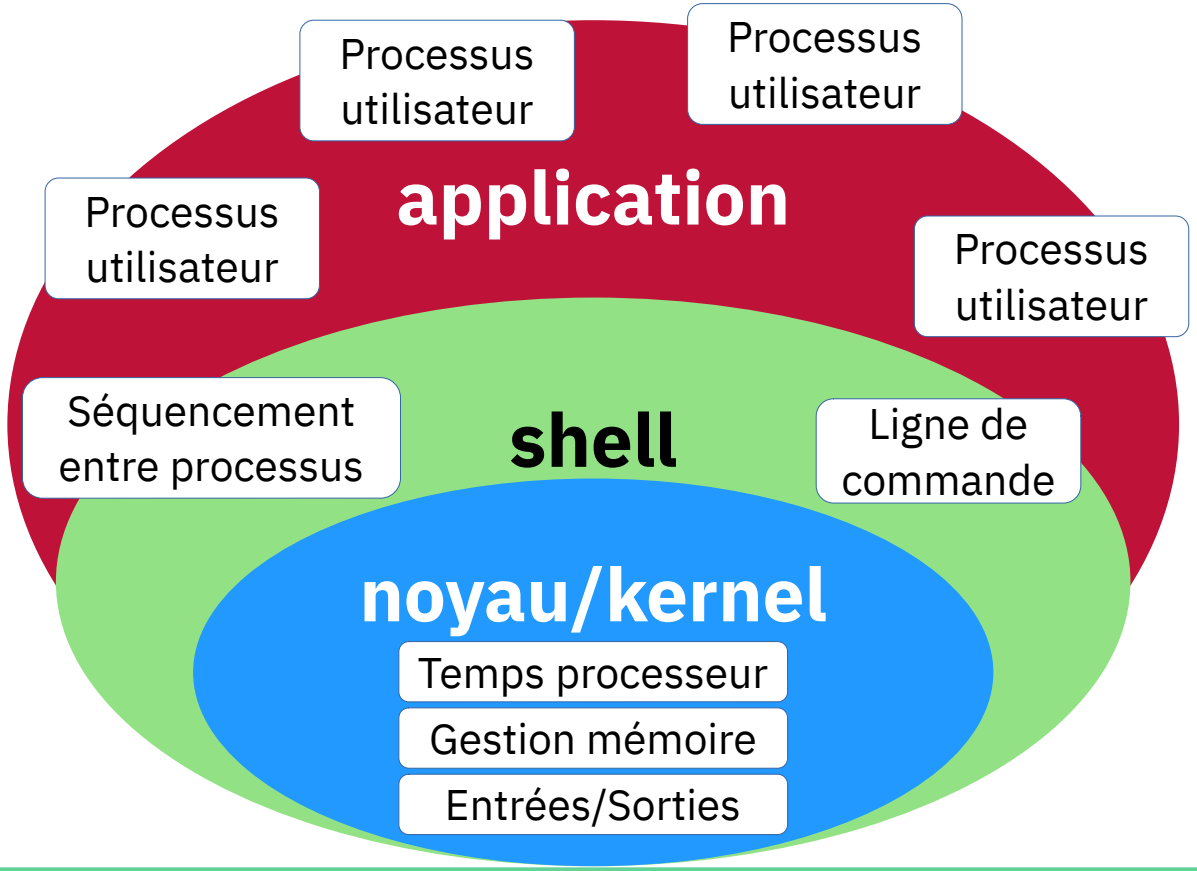
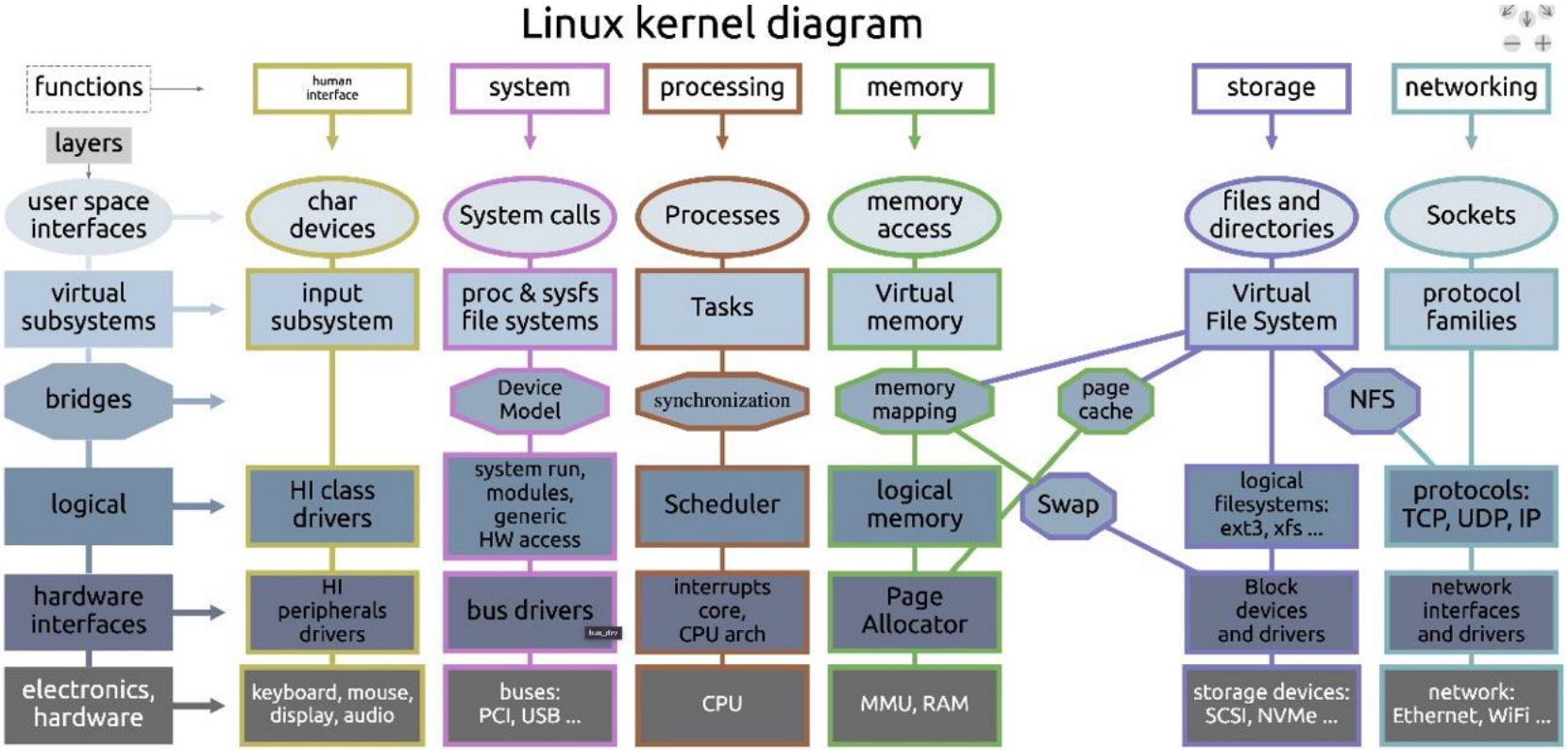


Diagramme du noyau Linux



Source : Constantine Shulyupin, makelinux.github.io/kernel/diagram

Définir la notion de mémoire virtuelle

- Un espace d’adressage propre à chaque processus
  - Adresses virtuelles traduites en adresses physiques
  - Isolation : un processus ne voit pas la mémoire des autres
- Pagination
  - Pages de 4 Kio décrites par des tables de pages
  - Traduction matérielle par la MMU du processeur
- Mémoire d’échange : swap, zswap, zram
- Observer
  - cat /proc/meminfo – pmap $$ – ps -o pid,vsz,rss,comm – slabtop

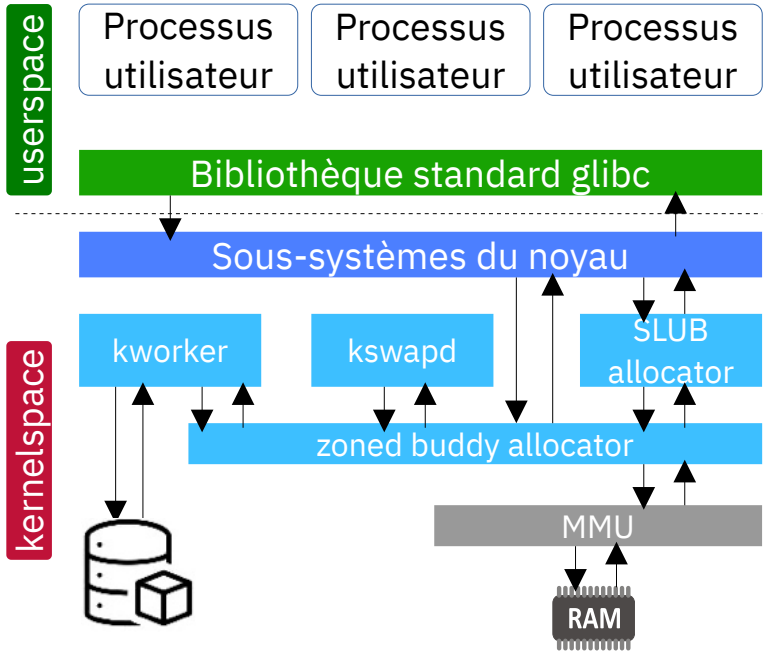
Mémoire virtuelle : les composants du noyau

userspace

- Programmes utilisateurs
- Bibliothèque standard glibc

kernelspace

- Zoned buddy allocator
  - Allocation des pages mémoire
- Slab allocator (SLUB)
  - Petits objets du noyau
- Threads noyau
  - kswapd, kworker (writeback)

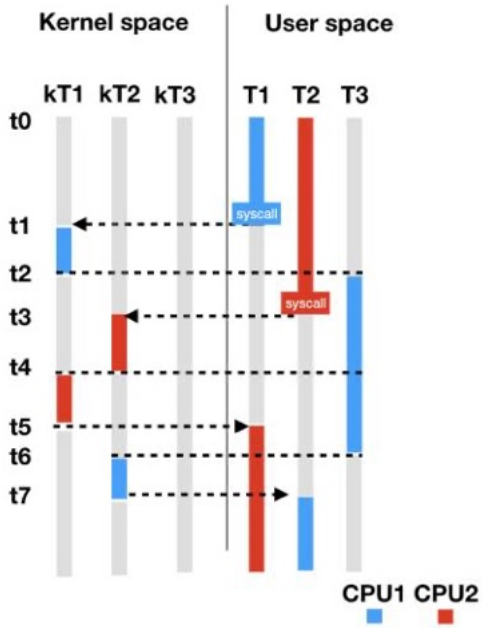


MMU → Memory Management Unit

Définir la notion d'ordonnanceur (*scheduler*)

3 domaines ou types de tâches

- Domaine temps réel
  - Contraintes de temps d'exécution
  - Fréquence d'exécution garantie
- Domaine entrées/sorties
  - Attente de disponibilité des périphériques
- Domaine CPU
  - Temps consacré aux calculs



Définir la notion d'ordonnanceur (*scheduler*)

Tranche de temps CPU (*time slice*)

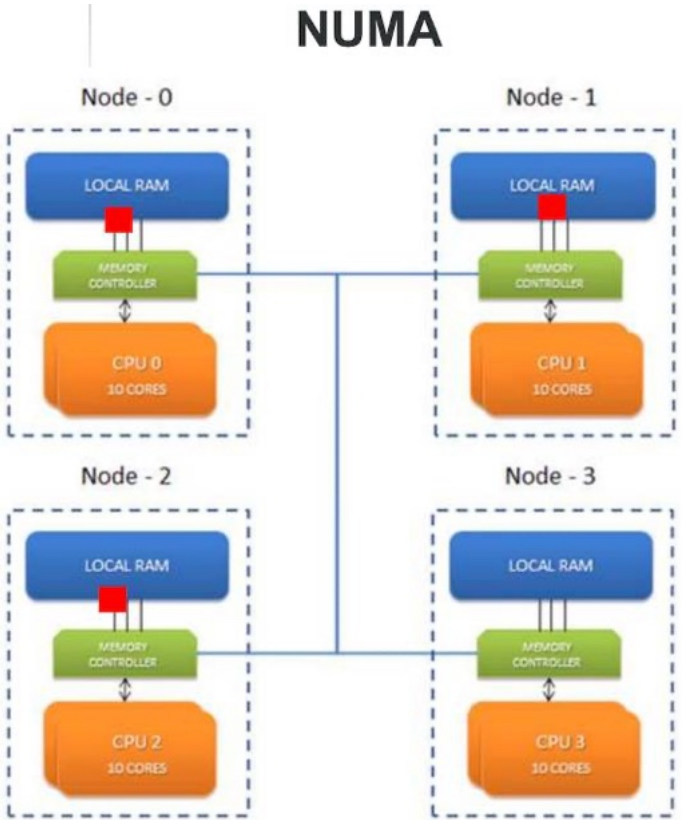
- Durée d'exécution d'un processus sur un cœur

Préemption

- Interruption d'un processus par un autre de priorité plus élevée

NUMA (*Non-Uniform Memory Access*)

- Placer la mémoire au plus près des processeurs
- Éviter les variations sur les temps d'accès



L’ordonnanceur Linux en 2026

- EEVDF remplace CFS depuis Linux 6.6 (2023)
  - Échéance virtuelle calculée pour chaque tâche
  - La tranche de temps devient une demande, plus une durée fixe
- Préemption
  - PREEMPT_RT intégré au noyau principal depuis Linux 6.12 (2024)
  - Linux 7.0 : modèles full et lazy, lazy par défaut
- sched_ext (6.12) : ordonnanceurs chargés en eBPF
- Observer
  - chrt -p $$ – taskset -p $$ – numactl --hardware

Bilan du cours 1

Retenir les points clés

Les systèmes GNU/Linux → 5 fonctions de base Unix

- Multitâches
- Multi-utilisateurs
- Portabilité
- Bibliothèques de développement standard
- Applications communes

Retenir les points clés

Une histoire continue depuis 1969

- 1969 : Unix aux Bell Labs, réécrit en C en 1973
- 1977-1978 : BSD à Berkeley, berceau de TCP/IP
- 1983-1984 : projet GNU et logiciel libre
- 1991 : noyau Linux → système GNU/Linux
- 2026 : Linux 7.x, Rust, Debian 13

Retenir les points clés

Le modèle en 3 couches d’un système GNU/Linux

- Noyau/kernel (gestion des ressources)
- Shell (interface utilisateur)
- Applications (processus utilisateur)

Décrire les mécanismes clés

Les mécanismes clés d’un noyau Unix/Linux

- Mémoire virtuelle
- Ordonnanceur (scheduler)
- Préemption
- NUMA (placement mémoire)
- Linux 2026 : EEVDF, préemption lazy, Rust