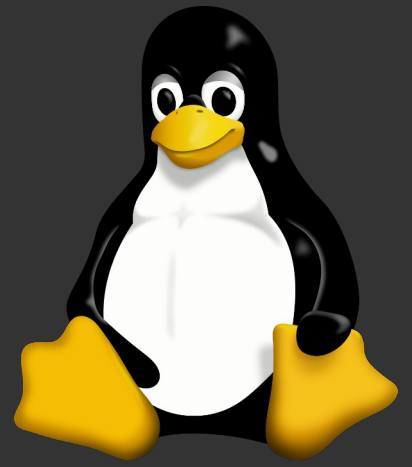
Linux
Fondamentaux



Damien Boureille - consultant système

Bienvenue

Merci de suivre notre formation.
Ces trois jours nous permettront de découvrir l'administration des systèmes Linux.

Après une présentation de Linux et ses spécificités, nous passerons en revue les commandes essentielles, le système de fichier standard, les utilisateurs, la gestion des processus et du réseau.

Nous vous souhaitons une excellente formation.

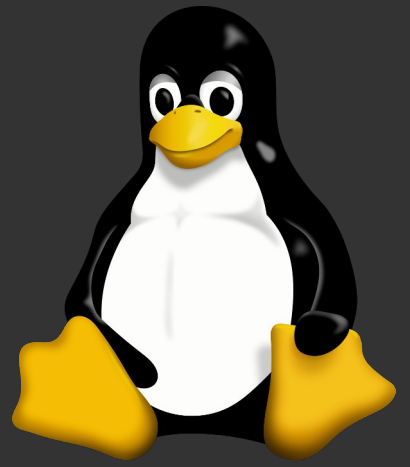
Présentons-nous

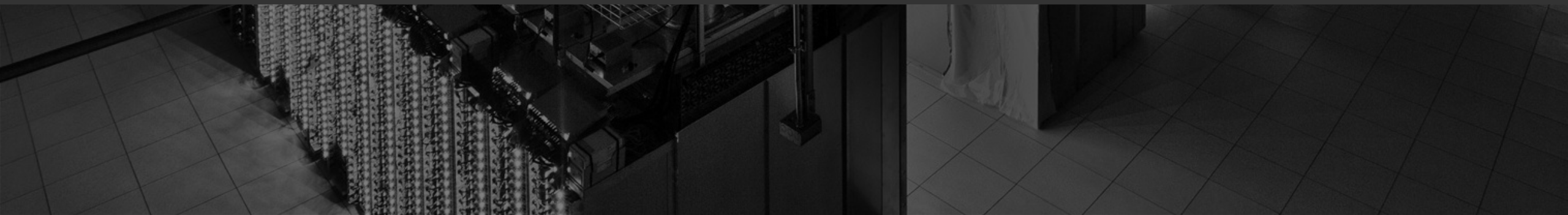
- Nom, société, fonction, expérience
- Connaissances préalables en administration Unix/Linux
- Attentes par rapport à ce cours

Plan de formation

- Fondamentaux
- Principaux outils
- Utilisateurs et sécurité
- Réseau et connectivité
- Gestion des processus
- Administration complémentaire

Linux
1 - Fondamentaux





Vue d’ensemble du module

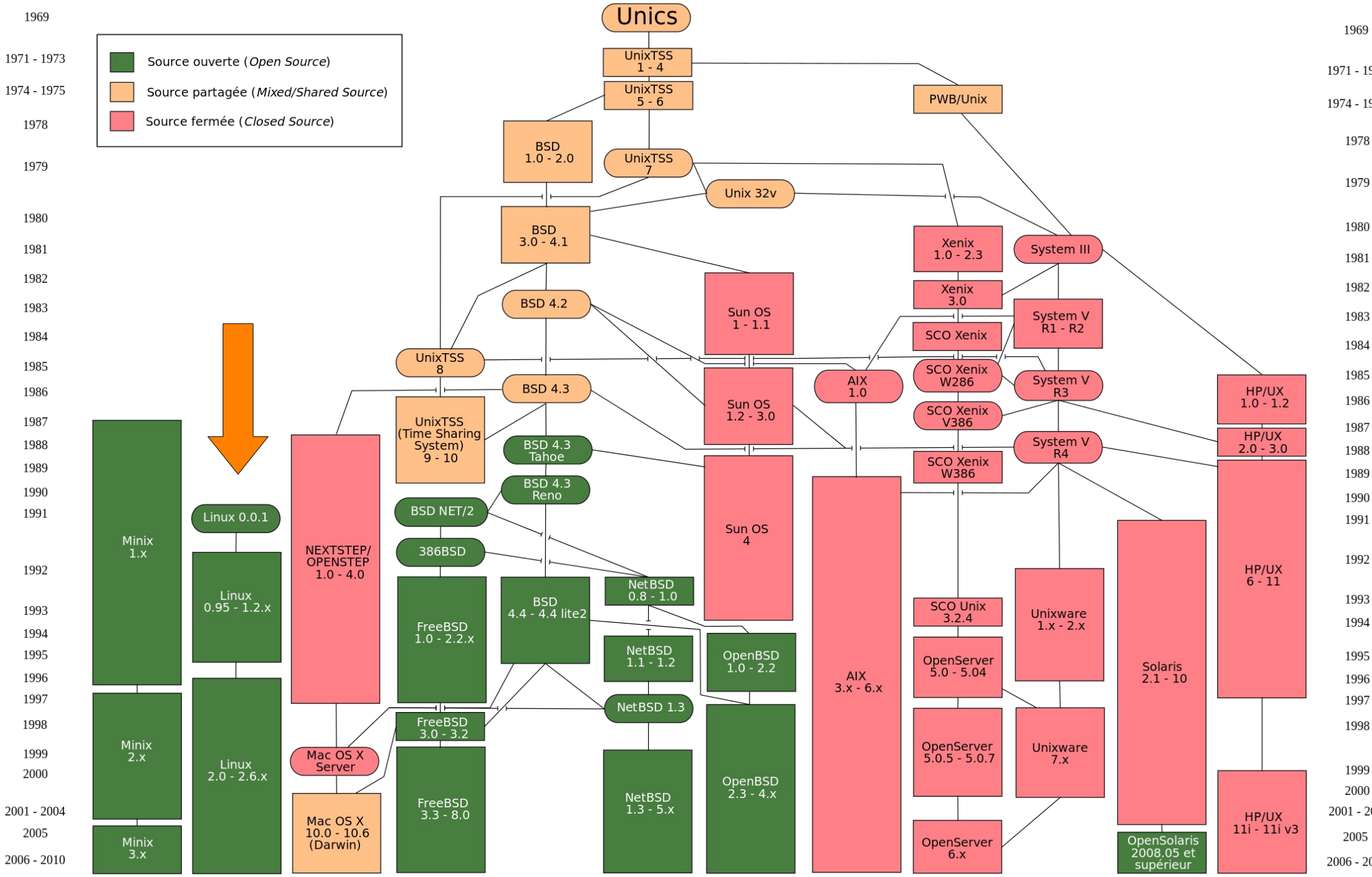
- Caractéristiques générales des systèmes Unix
- Spécificités de Linux
- Découverte du système de fichiers standard
- Console, terminaux, shells

Caractéristiques générales

- Unix est un système d'exploitation né à la fin des années 60 dans les laboratoires Bell, aux USA
- Simple par conception, à l'opposé des systèmes existant
- Approche modulaire plutôt que monolithique : fournir de l'outillage composable plutôt que de lourdes solutions complètes et partiellement redondantes
- Chacun de ses composants obéit à une règle : "Do one thing and do it well"

Caractéristiques générales

- A l'origine un système universitaire dans les années 70
- A partir de 1979, Bell fournit une licence Unix, qui devient grandement commercial dans les années 1980 : Solaris, AIX, HP-UX, DEC UNIX / Ultrix…
- Deux branches principales : System V et BSD
- La branche BSD deviendra open-source à partir de 1993 (NetBSD, FreeBSD, puis OpenBSD)
- En 1991, un étudiant finlandais écrit son propre noyau Unix pour matériel PC (Intel 386) : Linux


Source ouverte (*Open Source*)
Source partagée (*Mixed/Shared Source*)
Source fermée (*Closed Source*)
1969
1971 - 1973
1974 - 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 - 2004
2005
2006 - 2010
Unics
UnixTSS 1 - 4
UnixTSS 5 - 6
UnixTSS 7
UnixTSS 8
UnixTSS (Time Sharing System) 9 - 10
PWB/Unix
BSD 1.0 - 2.0
Unix 32v
BSD 3.0 - 4.1
BSD 4.2
BSD 4.3
BSD 4.3 Tahoe
BSD 4.3 Reno
BSD NET/2
386BSD
BSD 4.4 - 4.4 lite2
Sun OS 1 - 1.1
Sun OS 1.2 - 3.0
Sun OS 4
AIX 1.0
AIX 3.x - 6.x
Xenix 1.0 - 2.3
Xenix 3.0
SCO Xenix
SCO Xenix W286
SCO Xenix V386
SCO Xenix W386
SCO Unix 3.2.4
OpenServer 5.0 - 5.04
OpenServer 5.0.5 - 5.0.7
OpenServer 6.x
System III
System V R1 - R2
System V R3
System V R4
Unixware 1.x - 2.x
Unixware 7.x
Solaris 2.1 - 10
OpenSolaris 2008.05 et supérieur
HP/UX 1.0 - 1.2
HP/UX 2.0 - 3.0
HP/UX 6 - 11
HP/UX 11i - 11i v3
Minix 1.x
Minix 2.x
Minix 3.x
Linux 0.0.1
Linux 0.95 - 1.2.x
Linux 2.0 - 2.6.x
NEXTSTEP/ OPENSTEP 1.0 - 4.0
Mac OS X Server
Mac OS X 10.0 - 10.6 (Darwin)
FreeBSD 1.0 - 2.2.x
FreeBSD 3.0 - 3.2
FreeBSD 3.3 - 8.0
NetBSD 0.8 - 1.0
NetBSD 1.1 - 1.2
NetBSD 1.3
NetBSD 1.3 - 5.x
OpenBSD 1.0 - 2.2
OpenBSD 2.3 - 4.x

Caractéristiques générales

- *Stricto sensu*, Linux n'est qu'un noyau
- Le noyau est l'interface entre le matériel et les logiciels, il fonctionne dans un mode spécial du processeur, permettant un accès intégral au matériel
- Les logiciels accèdent aux ressources via des appels au noyau
- Le noyau permet aussi d'abstraire les spécificités du matériel : périphériques, processeurs…

Caractéristiques générales

- Si Linux n'est qu'un noyau, il faut donc un environnement utilisateur au dessus fournissant les services de base
- L'environnement **GNU** est généralement employé à cet effet
- GNU (GNU is Not Unix) est apparu au MIT au milieu des années 80
- Projet initié par Richard Stallman, le père du logiciel libre
- Fournit l'outillage de base du monde Unix au format libre, complémenté par des éditeurs, compilateurs, interfaces textuelles et graphiques…

Caractéristiques générales

- Le système complet, c'est à dire noyau Linux et partie utilisateur GNU, est dénommé **GNU/Linux**
- Par abus de langage (mais surtout par facilité), on dénomme généralement le système sous le seul nom de *Linux*
- A noter : d'autres environnements que GNU sont disponibles au dessus d'un noyau Linux : Busybox, Android, BSD…
- Inversement, l'environnement GNU peut fonctionner sur d'autres noyaux (par exemple, la distribution *Git For Windows*)

Caractéristiques générales

- L'écosystème Linux fonctionne en anarchie, c'est à dire sans gouvernance centrale. Ceci est également propre à Internet, mais est unique dans le monde Unix
- A la place d'une entreprise fournissant une feuille de route, chaque projet émerge donc de façon volontaire en fonction des besoins rencontrés, et croît de façon organique. Les projets pertinent perdurent, les projets non viables disparaissent
- De là, chacun compose un système adapté à ses besoins en fonction des composants disponibles

Caractéristiques générales

- Un système complet nécessite de choisir typiquement :
  - Une version du noyau Linux, à adapter soi même
  - Une libc (le runtime standard du langage C) : glibc, musl…
  - Un système d'amorçage : systemd, OpenRC, dinit, runit…
  - Un compilateur C : gcc, clang…
  - Un environnement utilisateur : GNU, Busybox
  - Une collection de services serveur ou utilisateurs
  - Autres…

Caractéristiques générales

- Une fois choisi chaque composant, il convient de récupérer le code, le compiler (le transformer en exécutable pour son matériel), l'installer, le mettre à jour…
- L'ensemble forme un travail considérable, à temps plein, et géré par des professionnels !

Caractéristiques générales

- Certaines organisations commerciales ou communautaires s'occupent donc ce travail à notre place
- Elles fournissent un ensemble prêt à l'emploi, possédant un installeur, une aide, un support, et établissant un standard
- L'ensemble forme ce que l'on appelle une **distribution**

Caractéristiques générales

- Quelques distributions majeures :
  - **Debian** (depuis 1993)
    - Communautaire à gouvernance démocratique, plus de 6000 contributeurs dont des développeurs noyau, des chercheurs en sécurité…
    - Surtout employée en environnement serveur
    - Pas de support officiel
    - Fréquemment rencontrée dans les cadres universitaires et institutionnels

Caractéristiques générales

- **Redhat** (depuis 1995)
    - Commerciale, rachetée par IBM en 2018
    - Le "Microsoft" de Linux
    - Onéreuse mais standard industriel
    - Distribution classique des clients grands-comptes

Caractéristiques générales

- Plusieurs dérivés non payant mais sans support de RHEL (Redhat Enterprise Linux) existent :
  - CentOS Stream
  - Rocky Linux
  - Alma Linux
  - Oracle Linux
- Chacune à ses spécificités, les quatre se rencontrent fréquemment en entreprise

Caractéristiques générales

- **Ubuntu** (depuis 2004)
    - Commerciale, distribution gratuite à support payant
    - Dérivé de Debian avec une emphase sur la facilité d'utilisation, la rapidité de mise en œuvre, le bureau
    - Possède un riche écosystème pour le cloud, les microservices et les objets connectés (IoT)
    - Standard pour les postes utilisateurs

Caractéristiques générales

- **Redhat**, **Debian** et **Ubuntu** forment les trois distributions principales du monde Linux
- Il en existe plus d'une centaine, notons notamment :
  - Alpine, minimaliste, pour l'embarqué, les VM, les conteneurs
  - Buildroot, Yocto, pour l'embarqué
  - Les distributions spécialisées (super-ordinateurs, embarqué…)
  - Arch, Void, pour les spécialistes et les développeurs
- NB : chacun (entreprise ou particulier) peut créer sa propre distribution

Arborescence

- Les systèmes Unix sont définis par des normes communes
- Par exemples, les appels systèmes du noyau sont normalisés dans le standard POSIX : Linux est un système Unix car il est "POSIX"
- Le système de fichiers et l'arborescence obéissent aussi à un standard : la **File Hierarchy Standard** (FHS)
- La commande `man hier` permet de la découvrir

Arborescence

- La FHS étant commune à tous les Unix, elle se retrouve aussi sur Android, iOS, les grands systèmes, etc.
- Exemple sur iOS →
- Découvrons cette arborescence…

Arborescence

`/bin`	Binaires système (commandes de base)
`/boot`	Amorceur, noyau
`/dev`	Périphériques
`/etc`	Configuration système
`/lib`	Bibliothèques système et modules noyau
`/home`	Dossier utilisateur

Arborescence

`/mnt`	Points de montage
`/sbin`	Binaires système superutilisateur
`/usr`	Programmes installés
`/var`	Hébergement de données (DB, VM…)
`/tmp`	Fichiers temporaires

Arborescence

- Certains dossiers sont spécifiques à certains Unix. Pour Linux :

`/opt`	Programmes **opt**ionnels, non gérés nativement
`/proc`	Interface avec les **proc**essus
`/run`	Sockets systemd
`/selinux`	Interface avec SELinux (systèmes Redhat)
`/sys`	Interface avec le noyau

- Egalement à savoir :

.	symbolise le dossier courant
~	symbolise le dossier personnel

Console

- La console est la sortie standard du noyau, ce que l'on voit au démarrage du noyau et/ou si on ne charge aucun programme graphique une fois démarré
- Historiquement un téléscripteur, aujourd'hui émulés sur les consoles système accessible via ctrl-alt-Fx
- Pas de souris, pas de défilement… simple et primitif, donc fonctionne tout le temps, même via un port série
- Peu ergonomique, on utilise plutôt un terminal

Terminal

- Un terminal est historiquement une machine externe autonome possédant un clavier et un écran, raccordé à l'unité centrale
- Aujourd'hui remplacé par des programmes graphiques nommés « émulateurs de terminaux », ou plus souvent simplement « terminaux »
- Possibilité d'utiliser la souris, sélection, copier-coller, barre de défilement…
- Console comme terminal permettent d'intéragir avec les programmes textuels tel que l'interpréteur de commande (shell), éditeur de fichiers et caetera.

Shell

- L’interpréteur de commande, le shell, est le programme affiché dans le terminal permettant d’interagir avec la machine
- Fonctionne en deux modes :
  - Interactif
    - Les commandes sont saisies, puis le résultat est affiché
  - Batch (script)
    - Les commandes sont saisies dans un fichier textuel, puis exécutées séquentiellement par le shell

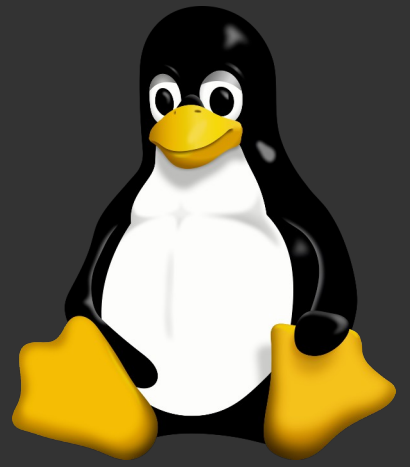
Shell - principes

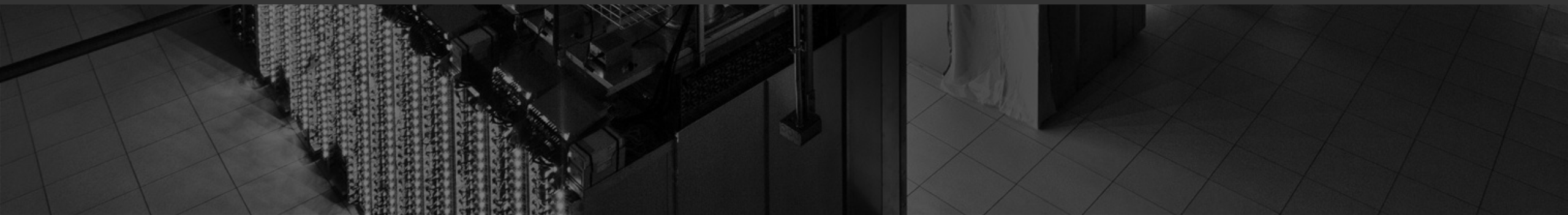
- Le shell remplit quatre fonctions fondamentales :
  - Fournir les commandes de base permettant de gérer le système d'exploitation
  - Vérifier la saisie de l'entrée avant de l'exécuter
  - Exécuter la saisie selon un mode déterminé
  - Permettre l'exécution de script (mode batch)
- Le mode de fonctionnement dépend du shell
- Sur Linux, le shell par défaut est généralement **Bash**. D'autres existent tel que ZSH, Busybox, Dash…

Shell - principes

- Un des principaux apports du shell, et à la base du fonctionnement d’Unix : la gestion des flux
- Comme nous l'avons vu : chaque programme ne fait qu’une seule chose, et la fait bien
- Le "workflow" typique du shell est composé d’une séquence de commandes traitant l’information
- Exemple :
  Lire le contenu de tel objet -> trier par ordre alphabétique -> ne conserver que les 5 premiers -> réaliser une opération sur ces 5 éléments

Linux
2 - Outils





Commandes - principes

- Dans le monde Unix, on évite les "mégacommandes" : on compose à la place des "phrases" de commandes nommées *lignes de commandes*
- Bash dispose d'un mécanisme d'auto-complétion des commandes, disponible en appuyant sur la touche [TAB]
- Penser à l'utiliser !
  - Gain de temps
  - Pas de faute de frappe

Commandes - système de fichier

- Opérations de base sur le système de fichiers :

`man`	aide sur une commande
`ls`	affiche le contenu d'un répertoire
`cd`	change de répertoire
`pwd`	affiche le répertoire courant
`mkdir`	crée un nouveau répertoire
`rm`	supprime un fichier, dossier
`cp`	copie un fichier, dossier
`mv`	déplace ou renomme un fichier, dossier

Commandes - système de fichier

- Autres opérations de base :

`cat`	affiche le contenu d'un fichier
`echo`	affiche l'argument sur la sortie
`grep`	recherche dans la sortie
`find`	recherche un fichier, dossier
`ps`	liste les processus
`kill`	envoi un signal à un processus

Commandes - système de fichier

- Les lignes de commandes comprennent des commandes, mais aussi de la ponctuation, composées de *métacaractères*
- Les métacaractères permettent de chainer les commandes et moduler leur comportement
- Les métacaractères portement par exemple sur des flux (redirections), des listes, des échappements, des expressions régulières…

Shell - métacaractères

- Voici la liste des principaux métacaractères :

  `>  <  *  ?  ` ' " | \ [ ] { } $ &`

- Chacun de ces caractères est interprété par le shell afin de modifier, contrôler, traiter, écrire ou récupérer de l'information
- Il est indispensable de les maîtriser, faute de quoi les lignes de commandes seront cryptiques

Shell - redirecteurs

< >

- Redirection de flux
- Permettent de rediriger la sortie vers un fichier, quel que soit son type
- Exemple :

```
echo 'Bonjour !' > ~/bonjour.txt
grep jour < ~/bonjour.txt
> ./nouveau-fichier
echo 'Ajout à la suite' >> ~/bonjour.txt
```

Shell - canaux

- Sous Unix, il existe trois canaux d'entrée / sortie standards :
  - **stdin** 0 Entrée standard
  - **stdout** 1 Sortie standard
  - **stderr** 2 Sortie des erreurs
- Par défaut, *stdout* (canal 1) reçoit toutes les sorties standards, tandis que *stderr* (canal 2), reçoit les erreurs
- Les deux canaux sont par défaut affichés dans le terminal

Shell - canaux

- < > permettent aussi de rediriger un canal dans un autre, ou dans un fichier. Par exemple, pour n'afficher que les erreurs :

```
/bin/command > /dev/null
```

- Autre exemple : ne rien afficher (silencieux) :

```
/bin/command >/dev/null 2>&1
```

- Détruire la sortie standard mais enregistrer les erreurs :

```
/bin/command 2>/var/log/erreurs >/dev/null
```

Shell - pipe

| (pipe, tube)

- Permet de rediriger la sortie standard d'un programme vers un autre programme
- Autrement dit, fait passer le canal 1 d'une commande au canal 0 d'une autre. |& transfère le canal 2 vers 0
- Essentiel pour la maîtrise du shell
- Exemple :

```
cat /etc/passwd | sort
rpm -qa | grep test
```

Shell - xargs

- xargs est une commande complémentaire du tube
- Là où le tube transfert la sortie de la première commande vers l'entrée de la deuxième, xargs transfert la sortie de la première commande comme *arguments* de la deuxième commande
- Exemple :

```
rpm -qa | grep test | xargs rpm -e
```

```
find . -name "*test*" | xargs du -h
```

Shell - backticks

` ` (backticks / accents graves)

- Permettent d'exécuter la commande saisie entre backticks et de la remplacer par son résultat une fois la commande exécutée
- Syntaxe alternative recommandée : **$(**commande**)**
- Exemple :

```
echo "Nous somme le $(date)"
rpm -i `rpm -qa | grep test`
```

Shell - jockers

* et **?**

- Caractères jokers
- * inclus toutes les combinaisons possibles de n'importe quelle suite de caractères
- ? inclus toutes les combinaisons possibles pour un seul caractère
- Exemples :

```
ls /etc/*.conf
ls /etc/rc?.d
```

Shell – double et simple quotes

" " et ' '

- Expressions littérales, le shell traite la saisie comme une seule séquence plutôt qu'un ensemble d'arguments
- " " indique que la séquence doit être évaluée, tandis que ' ' indique une séquence littérale, donc non évaluée par le shell.
- Exemples :

```
$ echo " Bonjour $USER  "
Bonjour damien
$ echo ' Bonjour $USER '
Bonjour $USER
```

Shell – échapement

\ (echappement)

- Permet d'indiquer au shell qu'un caractère faisant parti des métacaractères traditionnels ne doit pas être considéré comme tel. Très utile pour les caractères " et ', fréquemment employés
- Permet aussi de poursuivre une même commande sur une nouvelle ligne
- Exemple :

```
echo "\$USER: $USER"
USER: damien
```

Shell – listes et étendues

[ ] { } (étendues et listes)

- Permettent d'indiquer un ensemble de valeurs, ou de référencer les éléments d'un tableau
- Ainsi, l'étendue [0-9] indique tous les éléments de 0 **à** 9 inclus, tandis que la liste { 0,9 } indique seulement les éléments 0 **et** 9
- Exemple :

```
ls *.{jpg,jpeg,png}
```

Shell – listes et étendues

- Ainsi :

```
[a-dE16-9] = {a,b,c,d,E,1,6,7,8,9}
```

- Autre exemple :
  N'afficher que les éléments dont le nom contient au moins un chiffre :

```
ls ./*[1-9]*
```

Shell – contrôle

`&  &&  ||  ;`

- Contrôle des processus
- & exécute un processus en arrière-plan
- && exécute le deuxième processus si le premier a retourné vrai (0)
- || exécute le deuxième processus si le premier a retourné faux (! 0)
- ; exécute inconditionnellement le deuxième processus à la suite du premier

Shell – dollar

$

- Permet de lire une variable, indexer un argument, ou invoquer une opération interne à Bash
- Nécessaire pour le développement de scripts
- Exemple pour la création d'une variable :

```
$ message='Bonjour!'
$ echo $message
Bonjour!
```

Edition de texte

- Sous Unix, tout est fichier, et plus particulièrement fichier de texte
- Un éditeur puissant et minimaliste sert de standard : Vi (prononcé *vee-eye*)
- Pas le plus simple à apprendre, mais le plus efficace

Vi

- Vi est un éditeur de texte *modal* (deux modes d'utilisation)
- Initialement complexe à maitriser mais très efficace une fois appris
- Considérer vi comme un langage de mise en forme de texte, il est nécessaire d'apprendre ce langage pour apprécier vi
- Pour se mettre en selle rapidement : commande `vimtutor`

Vi

- Les deux modes sont le mode de commande (par défaut) et d’édition
- Le mode de commande est programmatique : permet de saisir une suite de commandes pour mettre en forme le texte.Exemple "couper du mot courant jusqu'à la fin du paragraphe, et le coller à la fin" → « d} Gp »
- Le mode d’édition permet de saisir et modifier le texte
- Passage en mode édition d’insertion : i
- Passage en mode édition d’ajout : a
- Retour en mode commande : [Echap]

Vi

- Quelques commandes :

Enregistrer :w (write)

Enregistrer sous :w /chemin/fichier

Quitter :q (quit)

Quitter sans enregistrer :q!

Quitter en enregistrant :wq

Aller à la ligne n nG (go)

Aller au début du document 1G (ou gg)

Aller à la fin du document G

Annuler u (undo)

Refaire ctrl-r (redo)

Vi

- Même logique pour copier : y (y$, yG, y5w…)
- Coller après le curseur p
- Coller avant le curseur P (majuscule)
- Annuler la dernière opération u
- Rechercher "mot" /mot
- Passer à l'occurrence suivante n
  Précédente occurrence N
- Rechercher le mot courant *
- Remplacer dans tout le texte :%s/non/oui/g
- Remplacer dans la ligne :s/non/oui/g

Vi

- Questions !
- Que fait la séquence suivante ?

```
^^ d2) {P
```

- Et celles-ci ?

```
wy$ gg 3p      $db^p
```

- Penser à pratiquer Vi quotidiennement jusqu'à l'acquisition des automatismes

Configuration du shell

- Chaque shell récupère automatiquement ses paramètres par défaut au démarrage
- Configuration commune à tous les utilisateurs (système): `/etc/profile`
- Par utilisateur :

```
~/.bash_profile
~/.bash_login
~/.profile_logout
~/.bashrc
```

Configuration du shell

- Il est parfois utile de créer un "raccourci" de commande
- Cela permet par exemple d'éviter la mémorisation d'argument, ou de créer des "métacommandes"
- La commande `alias` permet de créer un tel raccourci
- Sans argument, elle liste tous les alias existant
- Avec un argument, elle créé un alias
- Par exemple :

```
alias ls='ls -aF'
```

Configuration du shell

- Alias récupère sa configuration du fichier de profile, généralement .bashrc
- Pour être persistant, les alias seront ajoutés dans ce fichier
- Réciproquement, la commande unalias ainsi que le préfixe "\" suppriment un alias

Configuration du shell

- Le shell reconnait toutes les commandes se trouvant dans une variable nommée PATH (pour la voir : `echo $PATH`)
- Si le chemin d'une commande ne se trouve pas dans la variable PATH, il est nécessaire de la préfixer de son chemin complet
- Exemple, ajouter /usr/local/bin dans le chemin:

```
export PATH=$PATH:/usr/local/bin
```

Configuration du shell

- Pourquoi "export" dans la commande précédente ?
- PATH est une variable dite *globale*, ou encore d'environnement
- Elle est disponible à travers toutes les sessions plutôt que la seule session courante de l'utilisateur
- Export permet de propager une variable comme variable globale (d'environnement)
- Par convention, les variables d'environnements s'écrivent en majuscule

Configuration du shell

- Outre la variable PATH, il existe plusieurs constantes dans le shell pour désigner des répertoires :

  `.` dossier courant

  `..` dossier parent

  `~` dossier utilisateur

- Exemple:

  `ls /usr/local` et `cd /usr/local ; ls .`

  génèrent le même résultat

Configuration du shell

- PATH n'est pas la seule variable d'environnement
- D'autres variables sont utiles à connaitre :

  HOME emplacement du dossier personnel de l’utilisateur

  USER nom de l'utilisateur courant

  PS1 formatage de l'invite de commande
- Pour connaitre toutes les variables actuellement définies : commande `env`

Fichiers

- Une des caractéristiques des systèmes Unix (et leur successeur Plan9) est d'exposer toute l'information sous forme de fichier
- Un fichier n'est qu'un tableau de données, mais pas nécessairement enregistré physiquement dans une partition, il peut donc représenter n'importe quel lot de données y compris stockées en mémoire
- Il existe sept types de fichiers sous Unix

Fichiers

- Utiliser `ls` avec l'argument `-l` pour obtenir le type de fichier
- Le premier caractère indique si le fichier est un(e) :

d	dossier (directory)	**b**	périphérique bloc
l	lien symbolique	**p**	tube fifo (pipe)
c	périphérique caractère	**s**	socket
–	fichier régulier		

```
drwxr-xr-x 2 damien damien 4.0K  3 nov.  22:43 dossier/
-rw-r--r-- 1 damien damien    0  3 nov.  22:41 fichier
lrwxrwxrwx 1 damien damien    4  3 nov.  22:41 lien -> tube|
prw-r--r-- 1 damien damien    0  3 nov.  22:41 tube|
```

Fichiers

- Certains types de fichiers peuvent être inconnus des débutants :
  - Bloc : tableau de données de taille finie
  - Caractère : flux continu de données, que l'on peut lire au fur et à mesure de leur arrivée. Pas de cache avec le mode caractère.
  - Socket : expose le contenu d'une socket Unix dans un fichier
  - Pipe : expose le contenu d'un tube Unix dans un fichier
- La commande find permet de rechercher par type de fichier.
  Exemple :

```
find /dev/ -type b ;  find /run/ -type s
```

Systèmes de fichiers

- Un système de fichiers (FS) est une structure de données enregistrée dans une partition, permettant de retrouver les informations à l'intérieur de cette partition, tel un glossaire
- Les informations seront ensuite représentées sous forme d'arborescence (fichiers et dossiers)
- Joue le rôle d'une couche d'abstraction au dessus des secteurs et cylindres d'un disque
- FS par défaut sous Linux : ext2/ext3/ext4, ou plus récemment XFS

Systèmes de fichiers

- Un système de fichiers doit être monté pour être utilisé
- Cette opération se fait via la commande `mount`
- Peut être monté n'importe où, mais par convention dans le dossier /mnt pour les montages permanents, et /media pour les montages temporaires
- Un pilote du système de fichier doit être disponible pour que l'opération fonctionne
- S'il s'agit d'un système de fichiers exotique, vérifier la présence du module et le charger le cas échéant (`modprobe`). Il est parfois nécessaire d'installer un ou des paquets supplémentaires (exemple : nfs-common)

Systèmes de fichiers - gestion

- La commande mount est simple d'emploi :
  mount /dev/partition /mnt/cible
  exemple : `mount /dev/sdb1 /mnt/data`
- Pour rendre le montage permanent, on peut l'inscrire dans /etc/fstab
- Pour consulter tous les systèmes de fichiers montés :
  mount (sans argument) ou cat /etc/mtab
- Pour démonter un FS : `umount /mnt/cible`

Systèmes de fichiers - gestion

- La commande mount accepte des options via l'argument « -o » :

-a	monte tous les FS listés dans /etc/fstab
-o ro	monte le FS en lecture seule
-o rw	monte le FS en lecture-écriture (défaut)
-o noexec	ne permet pas l'exécution de binaires
-o noatime	ne met pas à jour les dates d'accès
-o nosuid	ne permet pas l'application de bit SUID

- Ces options sont également utilisables dans /etc/fstab

Systèmes de fichiers - gestion

- Dernières notes :
  - umount nécessite qu'aucun accès ne persiste sur le FS à démonter
  - Utiliser lsof ou fuser pour déterminer quels fichiers sont toujours en cours d'utilisation avant de démonter
  - Utiliser umount –l (lazy) pour détacher le FS et supprimer toute référence dès que le FS n'est plus occupé
  - Utiliser umount –f pour forcer un démontage. Parfois utile mais peu recommandé !

Systèmes de fichiers - gestion

- Nous avons vu que /etc/fstab automatise le montage de partitions, principalement lors du démarrage
- Syntaxe du fichier :
  volume, point de montage, FS, options, dump flag (sauvegarde), ordre de fsck
- Exemple :

```
/dev/sda1  /         ext4    defaults      0     0
/dev/sda2  /home     xfs     noexec        0     1
/dev/sda3  none      swap                  0     0
```

Systèmes de fichiers - gestion

- Certains types systèmes de fichiers doivent parfois être vérifiés manuellement (ex: ext2)
- Le volume ne doit pas être monté pour pouvoir être vérifié
- La vérification se fait avec la commande fsck. Exemple :

```
fsck.ext4 /dev/sdb1
```

- Utiliser –f et –y pour réparer automatiquement. Exemple :

```
fsck.ext4 –f –y /dev/sdb1
```

Systèmes de fichiers - gestion

- Il est enfin possible d'installer un système de fichier via la commande mkfs
- L'installation d'un système de fichier se dénomme « formatage », il cible toujours une partition et non un disque, que l'on partitionne plutôt que formate
- Exemple d'utilisation : `mkfs.ext4 /dev/sdb1`

Systèmes de fichiers - gestion

- Quelques autres commandes utiles :
- df (disk free)
- Principaux arguments :
  -T : type de FS
  -h : préfixes SI (KB, MB, GB…)
  -i : inodes

- lsblk list block devices
- blkid block ID

Systèmes de fichiers

- Un système de fichiers contient des entrées dénommées noeud d'information, ou inode
- Une inode est une entrée dans la table de données, et un pointeur vers la données concernée
- Enregistre également les métadonnées : propriétaire et groupe, permissions, taille, dates, etc.
- Consulter une inode : commande stat

```
stat /etc/fstab
```

Systèmes de fichiers

- Le système de fichiers débute par un superbloc, d'une taille de 8192 blocs
- Structure contenant la géométrie du disque, espace disponible, et surtout l'emplacement de la première inode
- Si le superbloc est endommagé et non réparable, le disque est irrécupérable
- Sous ext*, le superbloc est sauvegardé tous les 8192 blocs

ext2

- Système de fichier historique sous Linux, dérivé du système de fichier de MINIX
- Conception datée, mais minimal donc stable et performant
- Nécessite une vérification complète du FS lors d'un arrêt non prévu
- Limites selon la taille de bloc choisie :
  - Taille max. nom de fichier : 255 caractères
  - Taille max. fichier : 16GB-2TB
  - Max. fichiers sur FS : 10^8 (100 millions)
  - Taille max.volume : 2-32TB

ext3, ext4

- Ext3 apparait en 2001 et apporte la journalisation
- Conserve les limites de stockage d'ext2
- Ext4 : amélioration de ext3 (2008)
  - Repousse fortement les limites d'ext2/ext3 via un passage au 48 bits :
  - Taille max. fichier : 16TB
  - Max. fichiers sur FS : 4^9 (4 milliards)
  - Taille max.volume : 1EB (1024PB)

ext3, ext4

- Ext4 apporte aussi de nouvelles fonctionnalités, en particulier :
  - fallocate() : préallocation de blocs, c'est à dire réservation d'espace en amont afin de garantir la continuité des blocs alloués
  - allocate-on-flush : retarde l’écriture du bloc à un moment donné, généralement pour mutualiser les écritures
  - Somme de contrôle du journal
- Malgré les limites repoussées, non idéal pour des volumes > 100TB, où on lui préfèrera XFS

XFS

- Système de fichiers à 64 bits à haute performance et scalabilité linéaire
- Créé par Silicon Graphics (SGI) à partir de 1994 pour leur système Unix : IRIX
- Mêmes fonctionnalités que ext4, plus :
  - meilleure gestion des I/O parallèles via des groupes d'allocation
  - bande passante I/O dédiée à un processus
  - gestion native du stripping (RAID 0)
  - taille de bloc dynamique à la volée
  - attributs étendus
  - I/O direct (bypass du cache)
  - FS par défaut à partir de RHEL7

Btrfs

- Btrfs (prononcer B-tree FS) est basé sur les algorithmes B-Tree (d'où son nom) sur COW (copie sur écriture)
- Apporte le chiffrement natif, les sommes de contrôle, les instantanés (snapshots), le soft-pooling, le redimensionnement à chaud, les extents, la compression transparente en mode bloc (LZO, LZ4), le RAID logiciel dans toutes ses variantes (0,1,5,6, 10), la déduplication…
- Performant via l'apport du COW, et scalable via l'apport de l'agrégation de volumes (soft-pooling)
- Possibilité de conversion à chaud ext4->BtreeFS
- … mais "usine à gaz" pas toujours au point…
- Généralement non recommandé pour la production - favoriser son équivalent BSD : ZFS (Solaris, FreeBSD), plus mature

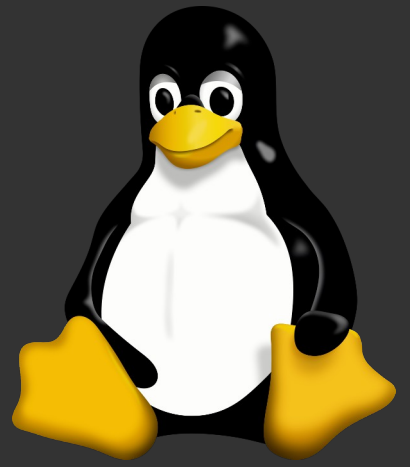
ZFS

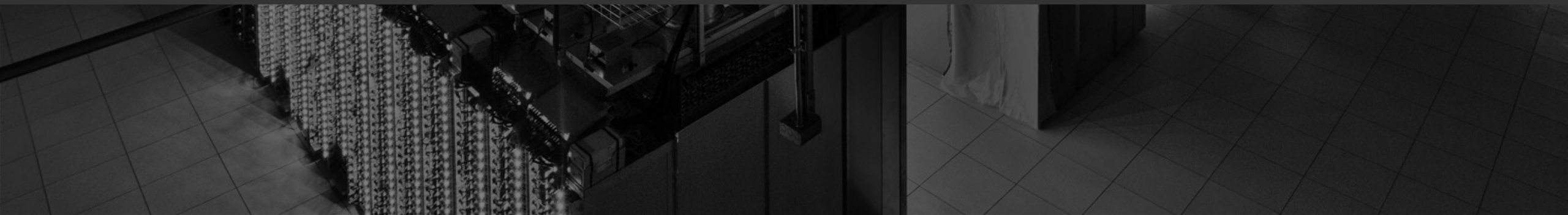
- Développé par Sun Microsystems (maintenant Oracle) depuis 2005
- Très haute capacité de stockage (128bits)
- Intégrations de tous les concepts de stockage connus
- Open-Source mais sous licence BSD, donc incompatible avec la licence GPL du noyau Linux
- Nécessite un pilote en espace utilisateur plutôt que noyau, d'où des performances dégradées sous Linux
- Plus fiable que Btrfs, mais là encore souvent assez lourd et complexe par rapport à la plupart des besoins

Que choisir ?

- De manière générale : ext4, le plus généraliste
- Si gros volumes de stockage : XFS ou ZFS
- Si besoin de fonctionnalités avancées : ZFS
- Pour l'interopérabilité : VFAT, XFAT et NTFS. XFAT est recommandé
- Nota bene : on tend aujourd'hui à créer des grappes de stockage, notamment via Ceph, plutôt que faire grossir verticalement les volumes

Linux
3 - Utilisateurs





Utilisateurs

- Sous Unix, chaque **u**tilisateur dispose d'un **U**ID et appartient à un ou plusieurs groupe(s)
- Les **g**roupes sont identifiés par un **G**ID
- L'accès à un objet (fichier, dossier, programme…)
  se fait selon l'UID et GID
- L'UID:GID zéro (root) possède un droit d'accès même sans être propriétaire ou avoir les droits de lecture

Utilisateurs

- Le fichier `/etc/passwd` centralise les utilisateurs. Il enregistre :
  - le nom de login utilisateur
  - le pointeur du mot de passe
  - l'UID et GID par défaut
  - le nom d'utilisateur
  - le dossier utilisateur
  - L'interpréteur de commande associé
  - Le premier champ (username) est le nom de login
- Les champs sont séparés par des caractères « : »

Utilisateurs

- Autrefois, les mots de passe étaient stockés dans `/etc/passwd`
- Désormais hachés dans `/etc/shadow` (en sha512 par défaut, paramétrable)
  - Structure d'une entrée shadow :
  - Login
  - Hash du mot de passe
  - Jours écoulés entre le 01.01.1970 et le dernier changement de mot de passe
  - Jours avant le prochain changement de mot de passe

Utilisateurs

- Structure d’une entrée shadow (suite) :
  - Jours avant d’avertir l’utilisateur du changement du mot de passe
  - Jours entre expiration du mot de passe et désactivation du compte
  - Jours écoulés entre le 01.01.1970 et la désactivation du compte
  - Exemple :
    `damien:$1$HEWdPIJ.$qX/RbB.TPGcyerAVDlF4g.:12830:0:99999:7:::`

Utilisateurs

- Le fichier /etc/group enregistre les groupes et leurs GID
- Structure d'une entrée :
  - Nom du groupe
  - Optionnel : mot de passe du groupe
  - GID
  - Membres, séparés par virgule
- Exemple :

```
wheel:x:4:damien,testuser
```

Utilisateurs

- Le compte root peut être utilisé directement (via login ou commande « su »), ou ses droits peuvent être octroyés à la demande à un utilisateur (commandes sudo ou doas, selon la distribution)
- Dans ce deuxième cas, les utilisateurs pouvant exécuter des commandes avec les privilèges root doivent être ajoutés au groupe « wheel »
- Avec systemd v256, sudo et doas sont dépréciés au profit d'une nouvelle commande : run0

Utilisateurs

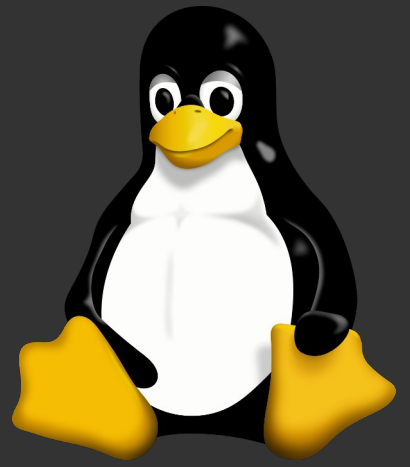
- La configuration générale de l'environnement utilisateur se fait via .profile et .bashrc
- Le fichier .profile est commun à tous les shells et surchage les paramètres par défaut du système, définis dans /etc/profile
- Le fichier .bashrc (ou .zshrc, etc.) est défini dans le dossier utilisateur par défaut : /etc/skel

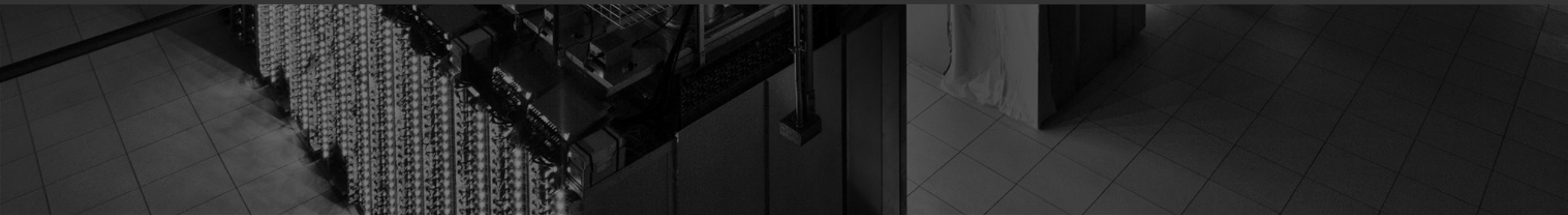
Utilisateurs

- Il est possible de créer un utilisateur via la commande `useradd`, complétée par les commandes `userdel`, `usermod`, `groupadd`, `groupdel` et `groumod`
- Il reste possible de créer un utilisateur manuellement :
  - Création de l'entrée utilisateur dans /etc/passwd
  - Création du mot de passe avec la commande passwd
  - Copie du dossier /etc/skel dans /home
  - Attribution des droits dans le dossier personnel

- Avec systemd, ceci devrait être remplacé par la commande `homectl`

Linux
4 - Réseau





Gestion du réseau

- Sous Linux, plusieurs sous-systèmes existent pour gérer le réseau
- Avant de les aborder, concentrons-nous sur le nommage des interfaces

Interfaces réseau

- Les interfaces réseau (NIC) possèdent une convention de nommage spécifique :
  - lo Boucle locale (loopback)
  - eth* Carte réseau ethernet
  - wlan* Carte réseau sans fil
  - ppp / pppoe Connexion 4G, RNIS/ISDN…
  - ath* Carte sans fil (chipset Atheros)
  - tap*, tun* Connexion VPN
- Exemple : /dev/eth0

Interfaces réseau

- La commande `ifconfig` permet de récupérer des informations relatives à chaque interface réseau, notamment son IP, son adresse physique (MAC/HWaddr) et les erreurs TCP
- Si aucune interface n'apparaît, vérifier que le noyau les détecte bien. Par exemple :

```
# lspci | grep Ethernet
00:03.0 Ethernet controller: Intel Corporation 82540EM
Gibabit Ethernet Controller (rev 02)
```

Interfaces réseau

- Si nécessaire, il est possible d'assigner un alias à une NIC, cela se fait via la configuration du module noyau correspondant :

```
echo "alias eth0   e1000" >> /etc/modprobe.conf
echo "alias eth1   r8169" >> /etc/modprobe.conf
```

- Rappel : la liste des modules réseaux disponibles pour un noyau se trouve dans /lib/modules/$(uname –r)/kernel/drivers/net

Interfaces réseau

- La gestion des interfaces se fait également via ifconfig
- Principaux arguments : up, down, netmask…
- Quelques exemples :

```
ifconfig  eth0      down
ifconfig  wlan0     up
ifconfig  eth1      broadcast  192.168.0.255
```

Interfaces réseau

- Autre exemple :

```
ifconfig        eth0       192.168.0.42
```

- Configuration complète:

```
ifconfig        eth0       192.168.0.42 netmask \
255.255.255.0 gateway 192.168.0.254

ifconfig        eth1       inet6 2001::DB8::3/64
```

- Ces changements sont perdus au redémarrage

Interfaces réseau

- Avec systemd, le nom des interfaces changent, de même que les commandes de gestion
- Les interfaces ne se trouvent plus immédiatement dans /dev comme avant (ex: /dev/eth0) mais dans /sys/class/net/
- ifconfig est remplacé par la commande ip, qui devient propre à Linux
- Usage :

`ip l`	Affichage des interfaces
`ip a`	Affichage de la configuration IP
`ip r`	Affichage des tables de routage
`ip netns`	Espaces de noms virtuels

Interfaces réseau

- Quelques exemples :

```
ip l set dev ensp31f6 down
ip a del 192.168.0.12/24 dev ensp31f6
ip a add 192.168.0.13/24 dev ensp31f6
ip r add default via 192.168.0.254 dev
ensp31f6
ip l set dev ensp31f6 up
```

Interfaces réseau

- Noter le nom des interfaces. Celui-ci possède sa propre convention de nommage
- Son principal atout est d'utiliser des alias déterministes : l'ordre de détection des interfaces par le noyau n'a plus d'importance
- Exemple :

```
$ lspci

(...)

00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit
Ethernet Controller (rev 02)

(...)
```

Interfaces réseau

- On remarque la détection d'une carte PCI, sur le premier bus au troisième emplacement (PCI 00:03)
- L'ancien "eth0" devient donc **E**ther**N**et **P**CI **0** **S**lot **3** : enp0s3
- Si la carte Wifi est sur le quatrième connecteur : WireLess PCI 4 Slot 3 - wlp4s3 (et caetera)
- Pour utiliser les noms traditionnels :
  - Argument `net.ifnames=0` au noyau
  - ...ou `ln -s /dev/null /etc/systemd/network/99-default.link`
  - ...ou surcharge du nom via une règle udevd

Interfaces réseau

- Pour rendre permanents une configuration réseau, il convient de l'inscrire dans un fichier de configuration
- Sur Debian et Alpine, il s'agit de /etc/network/interfaces
- Très simple de conception : même syntaxe que ifconfig !
- Exemple pour trois interfaces :

```
auto lo eth0 eth1

iface wlan0 inet dhcp

iface eth0 inet static
        address 192.168.0.1
        netmask 255.255.255.0
        gateway 192.168.0.1
```

Interfaces réseau

- Sur Redhat, plusieurs implémentations existent : network-scripts, Network-Manager et systemd-networkd
- Aucune n'est basée sur la syntaxe de ifconfig ou ip
- network-scripts est l'implémentation traditionnelle. Elle se configure dans /etc/sysconfig/network-scripts/ifcfg-NIC. Exemple pour deux interfaces :

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=10.0.1.27
USERCTL=no
```

```
DEVICE=eth1
BOOTPROTO=dhcp
ONBOOT=yes
DNS1=8.8.8.8
```

Interfaces réseau

- Les configurations pour Network-Manager et systemd-networkd sont plus complexes
- Pour Network-Manager : nmcli (man nmcli)
- La configuration de systemd-networkd passe par des unités de type network et un redémarrage du service
- Exemple d'une unité networkd :

```
[Match]
Name=eth0

[Network]
Address=192.168.1.10/24
Gateway=192.168.1.1
DNS=8.8.8.8
```

LAN

- La configuration du LAN se fait traditionnellement via trois fichiers :
  - `/etc/hosts` Résolution locale de nom
  - `/etc/networks` Nom des réseaux locaux
  - `/etc/resolv.conf` Serveurs de nom DNS
- Ces fichiers sont progressivement remplacés par systemd :
  - /etc/networks est remplacé par `systemd-networkd`
  - /etc/resolv.conf est remplacé par `systemd-resolvd`

Résolution de problème

- Remonter les couches plutôt que l'inverse :
  - La carte réseau est-elle détectée par le noyau ? (couche 1)
  - La carte réseau apparait-elle dans /dev ? (couche 2)
  - La carte réseau apparait-elle dans la commande ip ? (couche 3)
  - Peut-on lui affecter une configuration réseau et « pinger » une passerelle ? (couche 3)
  - Peut-on effectuer une requête curl sur un service réseau opérant ? (couche 4)

Résolution de problème

- Attention, systemd remplace bon nombre de commandes de diagnostique autrefois connues :
  - ifconfig → ip
  - netstat → ss
  - telnet → curl / nc (netcat)
  - traceroute → tracepath

Services réseau

- Le réseau opérationnel, nous allons maintenant passer en revue plusieurs services réseau :
  - SSH, permettant de se connecter à une autre machine
  - Rsync (remplaçant de scp), permettant de transférer des fichiers d'une machine à une autre
  - NFS et SAMBA, permettant de partager des fichiers via des lecteurs réseau

SSH

- Des commandes mentionnées, SSH est la plus simple d'utilisation
- Sa syntaxe est la suivante :

```
ssh USER@HOST  [commande]
```

- [commande] est optionnel, par défaut, ssh ouvre une nouvelle session distante
- Exemple d'utilisation : `ssh damien@192.168.0.1`
- Le port SSH est tcp:22, modifiable via l'option `-p`

rsync

- Rsync suit la syntaxe de SSH mais ajoute les chemins source et de destination :
  - Réplication de /opt/test vers [host]:/opt/test :

    ```
    rsync -avP /opt/test USER@HOST:/opt/test
    ```
  - Réplication de [host]:/opt/test vers /opt/test local :
  - `rsync -avP USER@HOST:/opt/test /opt/test`

rsync

- Rsync comprend un grand nombre d'options, consultables sur le man (`man rsync`)
- Notons néanmoins les options suivantes :
    - -a Mode archive (conserve tous les attributs des fichiers)
    - -z Compression
    - -v Verbeux (détaille les opérations)
    - -P Barre de **p**rogression

Services réseau - SAMBA

- SAMBA implémente le protocole SMB/CIFS et permet de se connecter à un groupe de travail ou un domaine Active Directory Microsoft
- Depuis SAMBA 4, il est également possible de créer un contrôleur de domaine maitre, ce qui n'était auparavant pas possible

Services réseau - SAMBA

- Principales fonctionnalités :
  - Support NetBIOS et WINS
  - Support de SMB2 et SMB3
  - Partage de fichiers
  - Partage d’imprimantes
  - Intégration dans un domaine
  - Contrôleur de domaine avec rôles FSMO

Services réseau - SAMBA

- L'architecture de SAMBA est client-serveur
- Serveur composé de deux services :
  - smbd : moteur, fournit les services d'authentification et d'accès aux ressources
  - nmdb : interfaçage avec TCP/IP, permet l'affichage des stations SMB dans le voisinage réseau et la découverte réseau
- Le client est invoqué par la commande smbclient
- D'autres commandes existent aussi : testparm, smbpasswd

Services réseau - SAMBA

- SAMBA s'installe lui aussi par les paquets de la distributions
- Selon que le poste est client ou serveur : samba-common, samba-client
- Un méta-paquet « samba » regroupe généralement l'ensemble et est à favoriser
- Une fois installé, il convient de le configurer en ajoutant les utilisateurs et les partages réseau, puis démarrer le service

Services réseau - SAMBA

- Le fichier de configuration principal se trouve dans /etc/samba/smb.conf
- Penser à réaliser une copie de sauvegarde de l'original avant de réaliser des modifications
- Trois sections principales :
  - Global - Paramètres généraux de Samba
  - Home - Configuration des dossiers personnels
  - Printers - Configuration des imprimantes

Services réseau - SAMBA

- Variables de configuration :

%a	Architecture du client, ex: WinNT, Samba
%I	IP du client
%m	nom NetBIOS du client
%M	nom DNS du client
%u	nom d'utilisateur
%g	groupe de l'utilisateur
%H	dossier personnel de l'utilisateur
%S	nom du partage
%T	date et heure courante

Services réseau - SAMBA

- Section [global], principaux paramètres :
  - Netbios name Nom du serveur Samba dans NetBIOS
  - Invalid users Liste noire, ex : root
  - Interfaces Choix des interfaces autorisées
  - Security Mode d'authentification (user/share)
  - Workgroup Groupe de travail à rejoindre
  - Server string Description du serveur à afficher
  - Domain master Active le rôle FSMO master
  - Hosts allow [EXCEPT] Hôtes autorisés
    (ex: hosts allow 192.168.0. EXCEPT 192.168.0.254)

Services réseau - SAMBA

- Section [home], principaux paramètres :
  - Comment — Affiche une description
  - Valid users — Utilisateurs autorisés
  - Browsable — Permet la lecture de la ressource pour tous
  - Read only — Interdit l'écriture sur le partage
  - Guest ok — Permet la connexion en tant qu'invité
  - Create mask — Masque de création, si non spécifié

Services réseau - SAMBA

- Section [printer], principaux paramètres :
- Comment — Description de l'imprimante
- Path — Chemin d'accès (ex: /dev/lp0)
- Guest OK — Autorise l'impression en tant qu'invité
- Printable — Active ou désactive l'imprimante
- Browsable — L'imprimante apparait dans l'explorateur de réseau

Services réseau - SAMBA

- Exemple de partage :

```
[Data]
comment = Données partagées
path = /mnt/data
browsable = yes
writable = yes
create mask = 0750
```

- Penser à utiliser la commande testparm pour valider la configuration

Services réseau - SAMBA

- En mode de sécurité « user », chaque utilisateur doit être authentifié
- Les comptes doivent être créés sur le système, ET dans Samba
- Comptes stockés dans le fichier /etc/smbpasswd
- Exemple d'ajout de l'utilisateur user1 :

```
useradd user1
passwd user1
smbpasswd -a user1
```

Services réseau - SAMBA

- Deux possibilités :
- Utilisation de smbclient (en voie d'obsolescence)
- Montage du partage avec mount (recommandé)
- Démonstration avec mount :

```
# mount –t cifs   //serveur/partage /mnt/partage  -o username=user1
# umount /mnt/partage
```

Services réseau - NFS

- NFS, Network File System, est le protocole de partage réseau natif Unix, développé à partir des années 1980 par Sun Microsystems pour SunOS puis Solaris
- Sa configuration est comparativement simple par rapport à SMB
- La principale différence concerne les droits, par station plutôt que par utilisateur
- Il est également possible de spécifier, optionnellement, des droits Kerberos

Services réseau - NFS

- L'installation se fait en une commande :

```
sudo apt install nfs-server
```

- Les partages réseau se définissent ensuite dans /etc/exports :

```
/var/partage/ *(rw,sync,no_root_squash,subtree_check)
```

- Consulter `man nfs` pour obtenir le détail des options
- On peut ensuite redémarrer le service nfs via systemd

```
systemctl restart nfs-server.service
```

- Et enfin vérifier que le partage est visible :

```
sudo exportfs -v
```

Services réseau - NFS

- Le partage peut ensuite être monté sur les hôtes cibles
- Tout d'abord installer le client nfs :

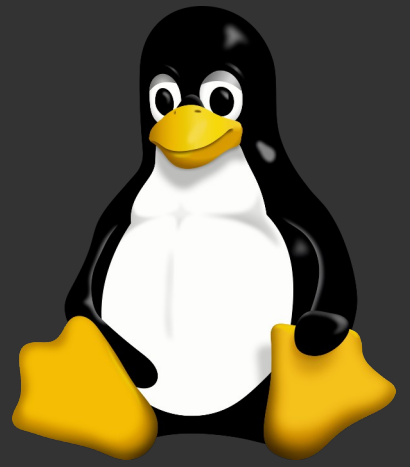
```
sudo apt install nfs-common
```

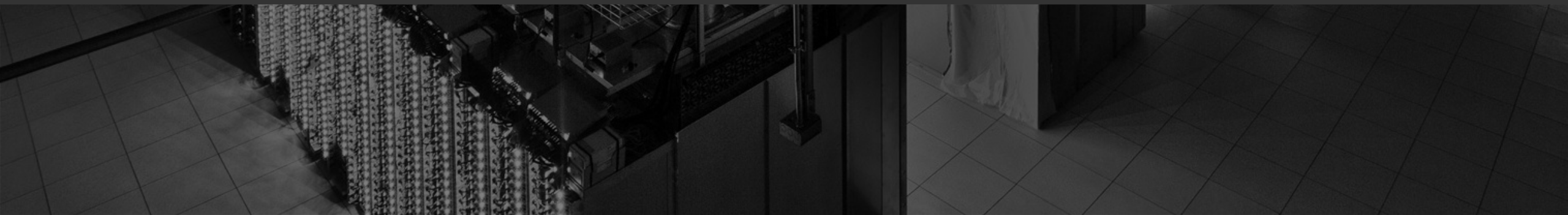
(ou `sudo yum install nfs-utils`)

- Monter ensuite le partage réseau localement :

```
sudo mount -t nfs  192.168.0.1:/var/partage
/mnt/nfs
```

Linux
5 - Processus





Processus

- Sous Unix, chaque processus est identifié par un ID : le PID
- Afficher les PID des processus : ps et pstree
- Sortie verbeuse : `ps axu`
- Pour pstree, à utiliser de préférence avec les options –p (PID) et –u (user) : `pstree –up`

Processus

- Les processus lancés depuis le shell sont visibles via la commande jobs, chaque job recevant également un ID d'identification (JID), préfixé du symbole « % ». Exemple :

```
$ jobs -l -p
1903
[1]+  Done        processus_en_cours
```

- Depuis le shell, le processus est adressable tant par son PID que son JID
- Exemple : `kill %1` ou `kill 1903`

Signaux

- Chaque processus Unix réagit à des signaux
- Permet de contrôler le processus en lui envoyant des messages
- Liste des signaux disponibles : `kill -l`
- 64 signaux différents, dont 32 (1 à 31) qui font parti du standard Unix (POSIX), et 32 (33 à 64) pour la gestion du temps réel, propres à Linux
- Envoyer un signal :

```
kill -NUMSIGNAL  PID
```

Signaux

- On peut ainsi terminer un processus en lui envoyant un signal 15 (SIGTERM) via son PID :

```
kill   -15  $(pgrep PROCESSUS)
```

- Si le processus ne peut répondre, on peut lui envoyer un signal 3 (SIGQUIT), voire 9 (SIGKILL) en dernier recours
- Les signaux les plus utiles sont SIGHUP (1), SIGINT (2), SIGQUIT (3), SIGKILL (9), SIGALARM (14), SIGTERM (15), SIGSTOP (19) et SIGCONT (18)
- Attention : l'ordre de SIGCONT et SIGSTOP (18 et 19) est inversé sur Linux par rapport aux autres Unix, notamment macOS

Top

- Top affiche combine les informations de vmstat, loadavg et ps
- Permet aussi le tri par consommation de ressource (CPU, vmem, IO, uptime…), mais aussi signaux, nices, etc.

Top

```
top - 22:07:29 up 170 days, 7 min,  1 user,  load average: 0.07, 0.02, 0.00
Tasks: 109 total,   1 running, 108 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.2 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    976.5 total,    398.0 free,    193.6 used,    486.5 buff/cache
MiB Swap:   1072.0 total,   1049.5 free,     22.5 used.    782.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 328873 damien    20   0    9016   4532   2648 R   1.0   0.5   0:00.42 top
      1 root      20   0  167940   6600   4796 S   0.0   0.7   3:18.29 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:18.94 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_kthr+
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trac+
     13 root      20   0       0      0      0 S   0.0   0.0   5:38.27 ksoftirqd/0
     14 root      20   0       0      0      0 I   0.0   0.0  14:13.70 rcu_preempt
     15 root      rt   0       0      0      0 S   0.0   0.0   0:00.01 migration/0
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     17 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     18 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/1
```

Top

- Top utilise des commandes interactives :

T	trier par temps processeur total
M	trier par consommation mémoire
P	trier par utilisation de processeur
k	envoyer un signal à un processus (15 par défaut)
r	envoyer un renice à un processus
i	[affiche/n'affiche plus] les processus inactifs
u	afficher les processus de tous les utilisateurs
f	choisir les colonnes (fields) de données
W	enregistrer la configuration dans ~/.toprc (write)

- Pour plus de touches et d'informations… man top !

Top

- La colonne STAT renseigne sur l'état du processus :

R	Running - processus en run queue Consomme activement de la ressource (sy, us...)
I	Idle - en fil d'attente dans l'ordonnanceur Ne consomme pas de ressource
S	Sleeping - en attente (timer sleep), interruptible
D	Delayed - en attente (ex, d'une IO bloc ou réseau) Non interruptible
T	Traced - le processus a reçu un appel système ptrace En cours de débogage (ex, gdb)
Z	Zombie - le processus ou son parent a été tué, mais la structure du processus reste résidente dans le noyau

Supervision

- Il est nécessaire de superviser les ressources afin de diagnostiquer et anticiper les problèmes
- *Superviser tout le nécessaire, mais rien que le nécessaire*
- Principales ressources à surveiller :
  - Ressources de calcul - charge système (loadavg)
  - Ressources de stockage - capacité, IO et disponibilité
  - Ressources mémoire - capacité, IO du fichier d'échange
  - Ressources réseau - charge, latence

Supervision - vmstat

- La plupart de ces ressources sont observables avec une commande universelle et historique : **vmstat**
- Commune à tous les Unix, dont Linux
- Synthétise les informations processus, mémoire, bloc, système et CPU
- Description détaillée des champs : `man vmstat`

```
~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 15607876  53712 1410316    0    0    41    37  574  774  9  4 86  0  0
~$
```

Supervision - loadavg

- Sur les systèmes Unix, la consommation CPU instantanée est secondaire
- Le niveau de charge est prioritaire : ni trop bas (sous-utilisation), ni trop haut (sous-disponibilité)
- Se mesure avec la charge moyenne : loadavg (`man uptime`)
- Est calculée avec le nombre de processus en fil d'exécution de l'ordonnanceur CPU (run queue) et IO, moyenné sur 1, 5 et 15mn
- Affiche aussi le nombre total de processus en fil d'attente, et le PID du programme demandant le plus de ressources

Supervision - loadavg

- Exemple de sortie :

```
$ cat /proc/loadavg
1.87 1.25 2.03 1/548 4382
```

- Cette dernière minute, 1.87 processus ont consommé de la ressource, sachant que cette machine dispose de 4 CPU. La disponibilité du système est donc garantie jusqu'à une charge de 4. La machine est actuellement sous-utilisée
- Actuellement, 548 processus sont en "run queue", et un seul consomme activement de la ressource, il s'agit du PID 4382

Supervision - stockage

- Pour superviser les ressources de stockage : iostat et iotop

```
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sdb               0.27         2.33        19.26     128477    1060734
sda               1.02        45.43        60.70    2501512    3342556
dm-0              8.36        44.69        60.70    2460986    3342496
dm-1              0.03         0.21         0.00      11504          0
dm-2              1.09         0.35         8.39      19042     461944
```

- Connaître l'état de santé d'un disque : smartctl, du paquet smartmontools (lecture des indicateurs SMART)
- Connaître les partitions du système :
  cat /proc/partitions
- Connaître les fichiers actuellement en cours d'utilisation : lsof (list open files)

Supervision - réseau

- Connaître quels processus consomment la bande passante : nethogs (à installer séparément)

```
NetHogs version 0.8.0

  PID USER     PROGRAM                                DEV        SENT      RECEIVED
9790  root     /usr/bin/python                        eth0      15.676     575.143 KB/sec
9712  damien   /usr/lib64/firefox/firefox             eth0       0.000       0.000 KB/sec
?     root     unknown TCP                                       0.000       0.000 KB/sec

  TOTAL                                                         15.676     575.143 KB/sec
```

- Idem, en mode pseudo-graphique (curse) : iftop
- Connaître le taux d'utilisation par NIC : sar –n DEV
- Connaître les sockets et processus utilisant le réseau : ss

Init

- Sous Unix, `init` est le PID 1, le programme racine chargé de démarrer tous les autres programmes
- Sur les Unix de type System-V, init se charge aussi d'amener le système dans un état désiré nommé « runlevel »
- Les runlevels sont au nombre de 7 :
  0 : arrêt 1 : utilisateur unique ("mode sans échec")
  2 : production 3 : libre
  4 : libre 5 : libre
  6 : redémarrage

Init

- Exception : Redhat !

0 : arrêt	1 : utilisateur unique (« mode sans échec »)
2 : libre	3 : production (serveur)
4 : libre	5 : production (poste de travail)
6 : redémarrage	

Init

- Init est configuré via /etc/inittab
- Nota bene : Linux n’appartient plus à la famille System-V et n’utilise plus init aujourd’hui
- Les autres Unix ne l’utilisent pas forcément non plus, les BSD ont leur propres système, ainsi que Solaris et macOS

Init

- Sur un Linux traditionnel, le inittab se compose de lignes indiquant l'identifiant du processus choisi, le ou les runlevels concernés, et l'action demandée
- Prototype : `ident:0123456:Action:Command`
- Exemple : `cmd:5:sysinit:/usr/bin/cmd`
- Si nécessaire : `man inittab`

Services

- Les services ne sont pas gérés par init mais par un superviseur, anciennement **rc**, aujourd'hui **systemd**
- Le superviseur s'assure que les services fonctionnent, fournit un état général, et les redémarre si nécessaire
- Traditionnellement configurés par des scripts dans /etc/init.d
- Exemple :
  `/etc/init.d/httpd {start,stop,status}`
- Certaines distributions implémentent également une commande "service" pour démarrer, arrêter, et obtenir le statut d'un service.
  ex : `service apache start`

Services

- La commande chkconfig permet d'activer ou désactiver un service. Ex:
  ```
  chkconfig httpd off
  chkconfig --level 55 iptables on
  ```
- Il est également possible de le faire à la main via un script ou un lien symbolique vers le bon dossier (exemples: vers /etc/rc2.d, /etc/runlevels/default …) – c'est l'approche "zéro code" traditionnelle propre à Unix : le système de fichier *est* la commande

De init à systemd

- sysvinit, le système d'initialisation des Unix SysV (traditionnellement Solaris, HP-UX, AIX et Linux…) accusait son âge :
  - scripts archaïques
  - peu basé évènement, ne reçoit que les signaux POSIX ([1..32])
  - pas de syntaxe déclarative
  - pas de lancement simultané
  - pas de gestion des dépendances
  - pas de cgroups, pas d'API, etc.

De init à systemd

- Certains Unix ont donc implémenté un système plus moderne :
  - Solaris : SMF
  - macOS : launchd
  - Linux : upstart, systemd, s6, runit, openrc, dinit…
- systemd (solution de Redhat) s'est imposé comme nouveau standard, malgré des polémiques techniques et philosophiques (complexe, hégémonique et tentaculaire)
- systemd est inspiré de SMF et apporte des fonctionnalités plus modernes au système d'init
- Certaines distributions permettent de choisir son système de démarrage. La plupart imposent toutefois systemd. dinit est son concurrent le plus isofonctionnel.

De init à systemd

Systemd remplace aussi certains standards Unix :

- cron, at → unités timer
- Xinetd → unités socket
- Fstab → unités mount
- Réseau → systemd-networkd
- resolv.conf → systemd-resolved
- ntpdate → systemd-timesyncd
- rsyslog → systemd-journald
- Getty → systemd-getty-generator
- consolekit → logind
- udisks → udev

De init à systemd

- Chaque élément de démarrage systemd est constitué d'une ou plusieurs unités (unit)
- Chaque unité peut être :
  - Un service (.service)
  - Une demande de montage (.mount)
  - Un timer (.timer)
  - Un démarrage par socket (.socket)
  - Un état désiré (.target), qui regroupe plusieurs unités
  - Et bien d’autres encore

De init à systemd

- systemd s'administre avec la commande systemctl
- Commandes principales :
  - Lister les services :
    `systemctl -t service`
  - Démarrer/Arrêter un service :
    `systemctl start|stop service`
  - Activer/Désactiver un service au démarrage :
    `systemctl enable|disable SERVICE`

De init à systemd

- Les unités sont chargées par des liens symboliques depuis /etc/systemd/system
- Pointent par défaut vers /usr/lib/systemd/system
  Ce dernier est le registre contenant toutes les unités possibles
- Chaque unité possède un arbre de dépendance permettant le démarrage en parallèle
- Possibilité de créer le lien soi-même, ou d'utiliser systemctl

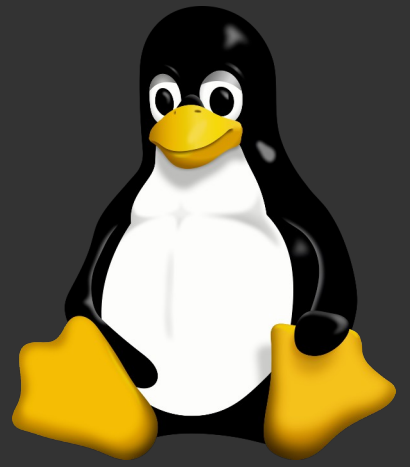
De init à systemd

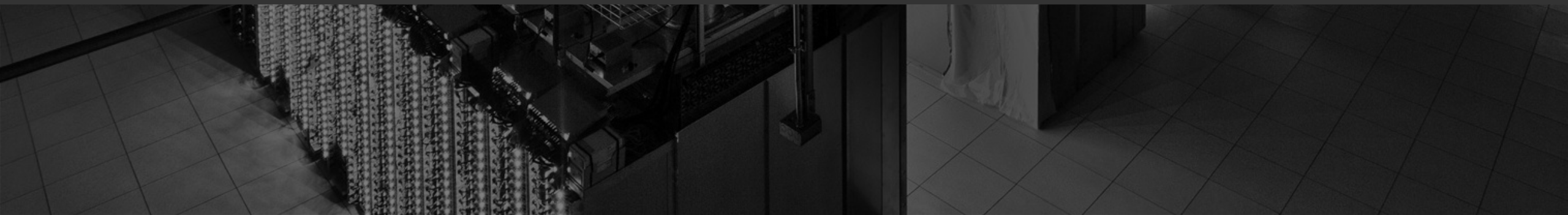
- La commande service est conservée à des fins de compatibilité
- Les anciens paquets non compatibles systemd disposent d’un mode de compatibilité sysV
- Ce dernier est géré via la même commande chkconfig
- Attention, chkconfig n’affiche et ne gère que les anciens services sysV, les services systemd n'apparaissent pas, et réciproquement avec les services sysV dans systemd !

De init à systemd

- On pouvait autrefois changer de runlevel à chaud : init 3, init 6 …
- Un mode de compatibilité systemd le permet encore, néanmoins les runlevels sont remplacés par les « targets »
- La commande `init 3` devient ainsi :
  `systemctl isolate multi-user.target`
- Pour éteindre le système, les commandes restent les mêmes : halt, poweroff, et idéalement `shutdown -h now` qui averti les utilisateurs et programmes.

Linux
6 - Compléments





Supervision - journaux

- Les journaux (logs) sont traditionnellement gérés par un service nommé (r)syslogd, mais parfois aussi socklog/nanoklogd, ou encore busybox
- Les logs sont par défaut enregistrés dans /var/log
- Journaux principaux :
  - Système / noyau : /var/log/messages
  - Démarrage (ring buffer) : /var/log/dmesg
  - Démarrage (runlevel) : /var/log/boot.log
  - Pour les journaux d'authentification :
    - /var/log/auth.log
    - /var/log/wtmp (ou commande last)
    - /var/log/secure

Supervision - journaux

- rsyslog se configure depuis /etc/rsyslog.conf
- Ce fichier permet de savoir quoi enregistrer, et à partir de quel niveau d'importance l'enregistrer
- Par exemple :

```
$ grep mail /etc/rsyslog.conf
*.info;mail.none;authpriv.none;    /var/log/messages
mail.*                             /var/log/maillog
```

Supervision - journaux

- rsyslog se configure avec trois paramètres : programme concerné, priorité, et chemin du log à enregistrer
- rsyslog peut capturer les évènements des sous-systèmes suivants : auth, daemon, cron, ftp, lpr, kern, mail, syslog, user
- Priorité : défini quel niveau rsyslog doit enregistrer comme évènement : Debug, Info, Notice, Warning, Error, Critical, Alert, Emergency
- Exemple : kern.warning /var/log/kernel.warning

Supervision - journaux

- Enfin, il est possible d'effectuer une rotation des journaux afin de purger les logs obsolètes, inutiles ou trop volumineux
- Un service dédié existe à cet effet : logrotate, configuré dans /etc/logrotate.conf
- Configuration de la fréquence : daily, weekly, monthly, yearly, et minsize
- Rotate x : indique combien d'anciennes versions de logs à conserver avant suppression

Supervision - journaux

- Exemple pour Apache : /etc/logrotate.d/httpd

```
/var/log/httpd/*log {
missingok    # ne retourne pas d'erreur si log absent
notifempty       # avertit si log vide
compress     # compresse les anciens logs avec gzip
postrotate       # exécute la commande suivante après
rotation
/sbin/service httpd reload > /dev/null 2>/dev/null ||
true
endscript
}
```

- Penser à consulter le man de logrotate

Supervision - journaux

- rsyslog et logrotate ont eux aussi été remplacés par systemd, qui possède sont propre service : journald
- Les journaux sont désormais au format binaire et nécessitent une commande dédiée pour les lire : journalctl
- Quelques exemples :
  - Derniers logs : journalctl -xe
  - Log du dernier démarrage : journalctl -b(oot)
  - Filtrer par date :
    - journalctl --since "2019-01-10" --until "2019-01-11 03:00"
    - journalctl --since yesterday --until "1 hour ago"

Supervision - journaux

- Autres arguments utiles :
  - Filtrer par service : `journalctl -u(nit) httpd.service`
  - Filtrer par PID : `journalctl _PID=1234`
  - Par chemin : `journalctl /usr/bin/bash`
  - Journaux du noyaux : `journalctl -k (ou --dmesg)`

Installation de programmes

- Historiquement, les paquets sont installés par code source :
  - Meilleure implication de la communauté
  - Meilleure qualité du code, mieux scruté, mieux testé
  - Plus de chance de partager des améliorations
  - Philosophie originelle du logiciel libre
- Mais fastidieuse, consommatrice de temps, nécessitant un niveau technique et impossible à faire à grande échelle
- D'où l'émergence des paquets binaires dans les années 1990

Installation de programmes

- Les paquets binaires sont compilés par l'éditeur (le créateur de la distribution) sur des serveurs de « build », puis copiés sur des dépôts, et enfin installés localement depuis ces dépôts
- Principaux outils de gestion de paquets :
  - Famille Debian : dpkg
  - Famille Redhat : rpm
  - Autres : pacman, xbps, apk, nix… (moins vus)

Installation de programmes

- Dpkg et rpm sont des outils bas niveau qui ne gèrent pas les dépendances, ni même l'installation depuis un dépôt
- Des surcouches ont donc été ajouté pour palier à ces inconvénients :
  - Famille Debian : **apt** (et aptitude)
  - Famille Redhat: **yum** (et dnf)
- Sur ces deux systèmes, ces commandes ont remplacé les anciennes qui ne sont plus utilisées que pour les tests, diagnostiques et réparation

Installation de programmes

- Comme toujours avec les systèmes Unix (y compris Linux), les commandes sont documentées dans des pages de manuel
- Faisons l'essai :

```
man yum
man apt
```

- Trouvez-vous comment appliquer les mises à jour avec ces deux commandes ?

Installation de programmes

- On tend aujourd'hui à rendre les déploiements déclaratifs plutôt que procéduraux : on définie un état désiré puis un orchestrateur s'assure de cet état sur des lots de machine
- Les outils de conteneurisation sont un classique
- Eventuellement : Ansible, Terraform (voire Puppet et Chef), mais ne sont pas immuables et ne garantissent pas autant la consistance du déploiement
- Sur Redhat, Satellite / Spacewalk permet d'administrer un parc de machine avec un tableau de bord web

Installation par sources

- Encore nécessaire aujourd'hui :
  - Développements spécifiques
  - Compléments « out of tree »
  - Besoin d'un programme en dernière version
  - Cross-compilation
  - etc.
- Egalement indispensable pour les développeurs

Installation par sources

- Fonctionne en plusieurs étapes :
  - Détection des dépendances
  - Génération d'un Makefile ou équivalent (recette de compilation)
  - Compilation
  - Installation
- Traditionnellement, ces outils sont autoconf et make. La procédure ci-dessus utilise alors ces trois commandes dans le dossier des sources :

```
$ ./configure
$ make
$ make install
```

Installation par sources

- D’autres outils ont été développés pour corriger certains archaïsmes de autoconf et make, notamment cmake, meson et ninja
- Généralement, les programmes récupérés depuis un gestionnaire de code source (ex: git) possèdent un README.md expliquant comment compiler le code
- La compilation est un grand classique des systèmes ouverts, ne pas hésiter à s’en servir