LINUX

Présentation

Fonctionnement

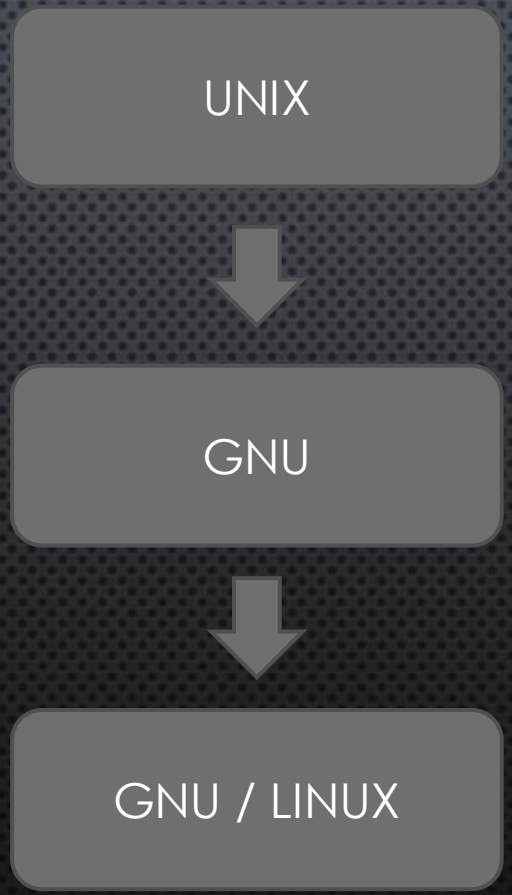
Commandes de base

Pascal MELOT - 30/12/2022

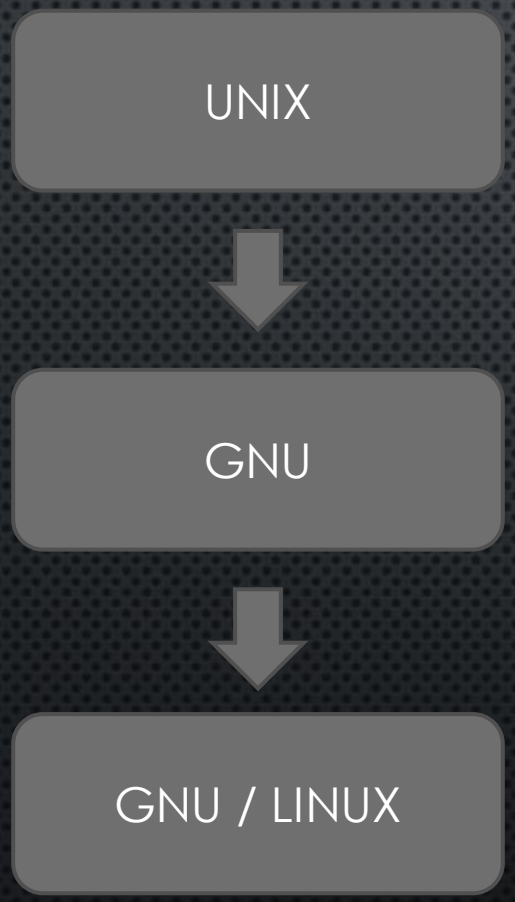
INTRODUCTION

UNIX, LINUX, GNU...

AVANT LINUX… UNIX



AVANT LINUX… UNIX



OS multitâche, multi-utilisateur
créé en 1969 par Kenneth Thompson & Dennis Ritchie (Bell Labs)
Création du Langage C

« GNU's Not UNIX » - OS libre qui reprend les concepts d'UNIX
lancé en 1983 par Richard Stallman

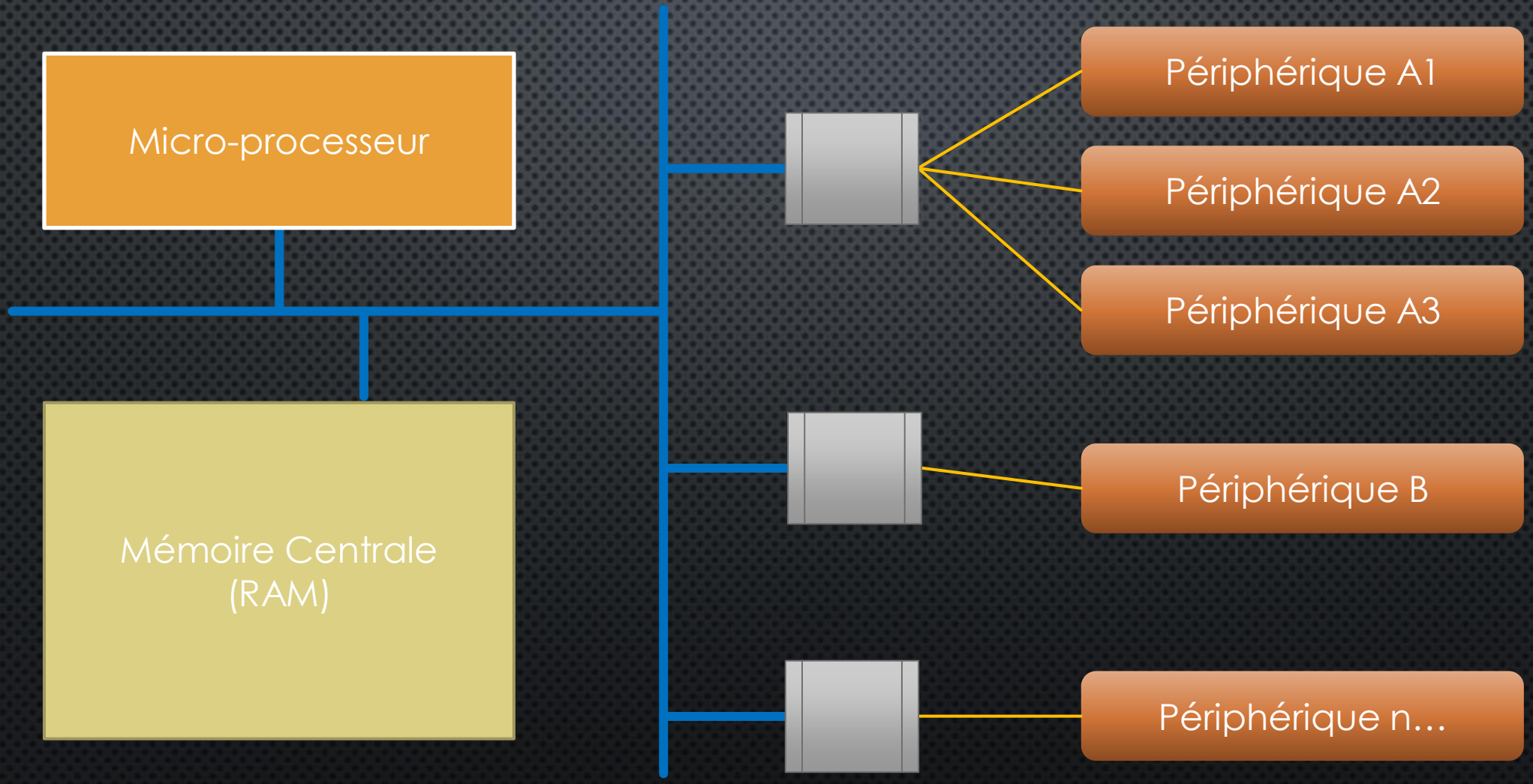
Noyau (kernel) pour l'architecture x86
Reproduit le comportement du noyau Unix originel (1969)
créé en 1991 par Linus Torvalds, intégré dans GNU

SYSTÈME INFORMATIQUE

Considérations sur le matériel, le logiciel

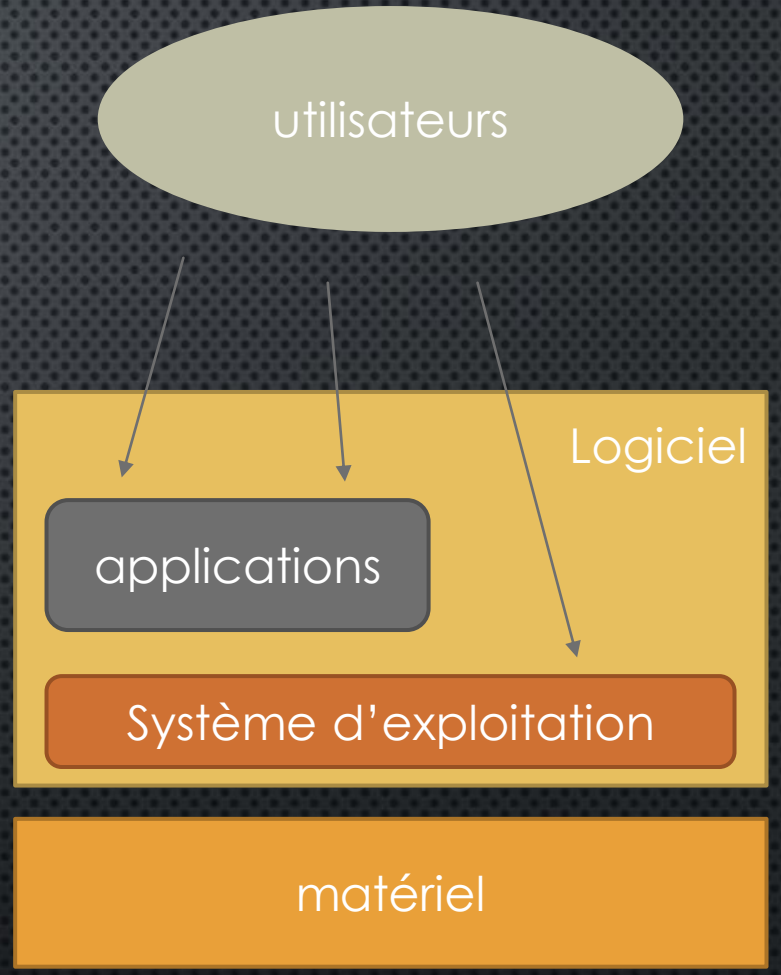
et le rôle d'un système d'exploitation

SYSTÈME INFORMATIQUE - MATERIEL



SYSTÈME INFORMATIQUE - OS

- Un système d'exploitation (Operating System) est un ensemble de programmes qui orchestre l'utilisation des capacités d'un ordinateur (ressources matérielles) par des logiciels applicatifs.
- Une des tâches de l'OS est d'offrir aux utilisateurs une interface simple et « conviviale » avec le matériel.



OPERATING SYSTEM

Transformer le matériel en une machine utilisable

Optimiser l'utilisation des ressources (matérielles et logicielles)

- SECURITE
- FIABILITE
- PERFORMANCE

FONCTION D'UN OS

- Le système d'exploitation s'occupe au minimum de :
  - La gestion des processus (programmes)
  - La gestion de la mémoire
  - Le système de fichiers
  - La gestion des entrées/sorties

- Un OS « moderne » est :
  - **Multi-tâches** : il permet d'exécuter, de façon apparemment simultanée, plusieurs programmes informatiques.
  - **Multi-utilisateurs** : il est conçu pour permettre à plusieurs utilisateurs d'utiliser l'ordinateur simultanément, tout en limitant les droits d'accès de chacun afin de garantir l'intégrité de leurs données.
  - **Multi-processeurs** : il est capable de répartir la charge de travail sur plusieurs processeurs, ou sur plusieurs cœurs de processeurs.

INTRODUCTION À LINUX

QU'EST-CE QUE LINUX ?

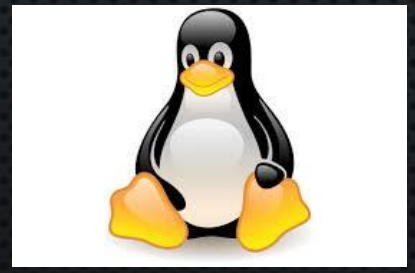
- Linux est d'abord le nom d'un noyau (le contrôleur central)
- Avec quelques outils supplémentaires, on obtient un système d'exploitation (OS) :
  - Un environnement Shell (une ligne de commande)
  - La gestion du système (ajouter des utilisateurs,...)
  - Des applications (mail, web, développement,...)
  - Le tout est mis dans une distribution Linux :
  - dépôts de paquetages, maintenance des logiciels, scripts de lancement,…
- interfaces graphiques, communautés, ...

ARCHITECTURES MATÉRIELLES

- Linux est supporté sur tout type d'architecture :
  - Serveurs d'entreprise
  - Serveurs de Data Center
  - Ordinateurs de bureau
  - Ordinateurs portables
  - Ordinateurs légers
  - Mainframes
  - Embarqués (Industrie, automobile, domotique, …)
  - Appareils mobiles, appareils légers
  - Périphériques d'infrastructure réseau/stockage/multimédia (appliances)

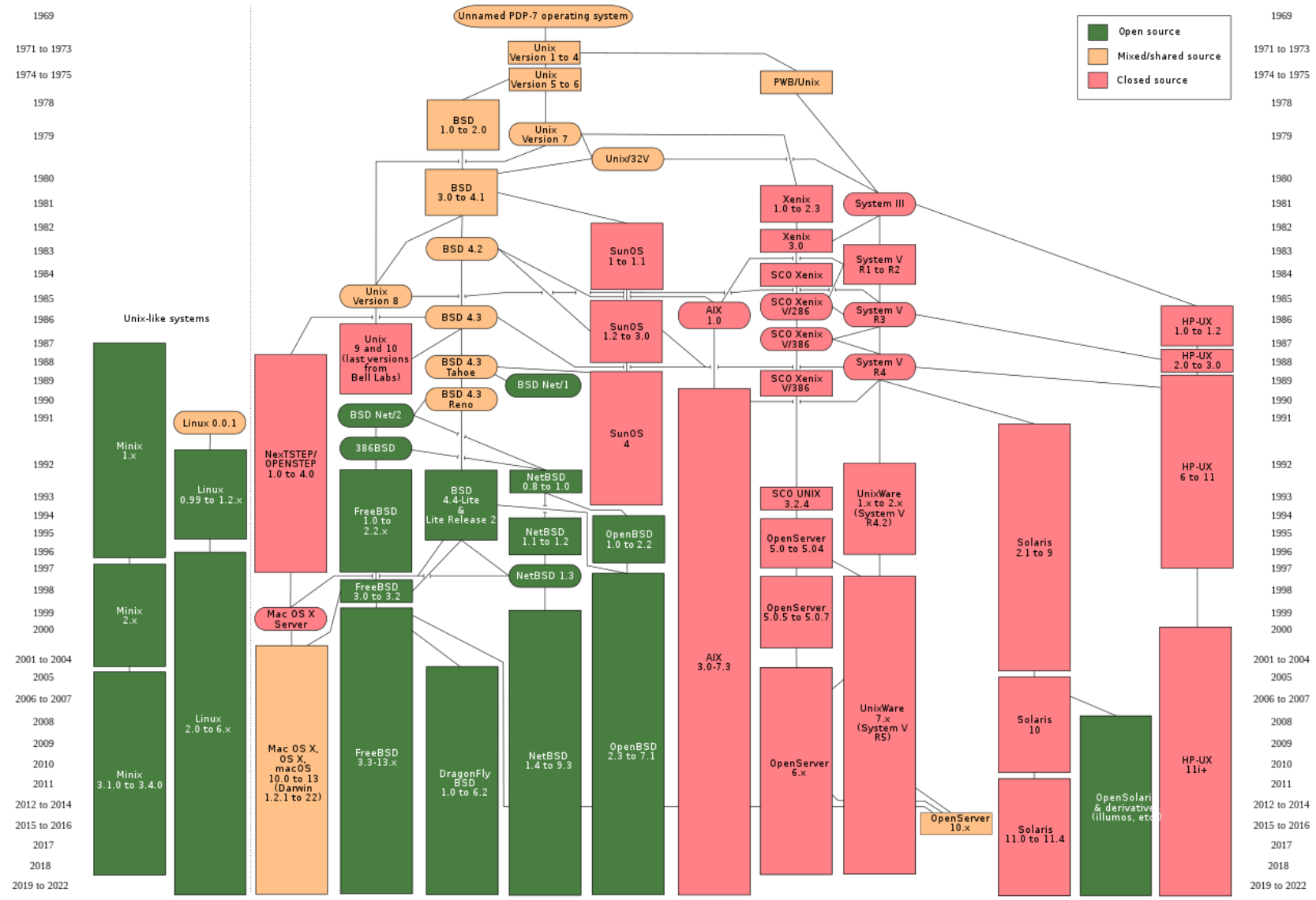
ORIGINE DE LINUX

- Créé en 1991 par Linus Torvalds pour des processeurs 80386,
- Inspiré par le projet pédagogique Minix.
- Reproduit le comportement d'un noyau UNIX (1969)
- Repris par une communauté de développement.
- Le projet GNU ajoute une série d'outils autour du noyau.



QU'EST-CE QUE UNIX ?

- Unix a été créé chez Bell Labs en 1969, par Ken Thompson & Dennis Ritchie.
- Populaire dans les milieux académiques et sur les Mainframes (1980).
- Donne le nom à une famille de systèmes d'exploitation (notamment FreeBSD, NetBSD, OpenBSD, Dalvik/Linux (Android), GNU/Linux, iOS et OS X.
- Le nom « UNIX » est une marque déposée de l'Open Group, qui autorise son utilisation pour tous les systèmes certifiés conformes à la *Single UNIX Specification*.



By Eraserhead1, Infinity0, Sav_vas - Levenez Unix History Diagram, Information on the history of IBM's AIX on ibm.com, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1801948

GNU ?

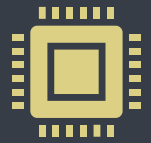
- GNU est un projet de système d'exploitation libre lancé en 1983 par Richard Stallman, puis maintenu par le projet GNU.
- Son nom est un acronyme récursif qui signifie en anglais « GNU's Not UNIX » (littéralement, « GNU n'est pas UNIX »).
- Il reprend les concepts et le fonctionnement d'UNIX.
- Le système GNU permet l'utilisation de tous les logiciels libres, pas seulement ceux réalisés dans le cadre du projet GNU.
- Il existe à ce jour deux distributions du système d'exploitation GNU :
  - Arch Hurd ;
  - Debian GNU/Hurd.

GNU / LINUX

- GNU/Linux (souvent appelé « Linux ») est une variante du système d'exploitation GNU fonctionnant avec le noyau Linux.
- Le projet GNU avait originellement prévu le développement du noyau Hurd pour compléter le système, mais au début des années 1990, Hurd ne fonctionnait pas encore et son développement rencontrait encore des difficultés.
- L'arrivée du noyau Linux permit l'utilisation du système GNU sur les ordinateurs animés par des microprocesseurs de la famille Intel x86, en favorisant sa large diffusion par la complémentarité des projets.
- Source : https://fr.wikipedia.org/wiki/GNU

EVOLUTIONS DE LINUX

Support de la virtualisation



Support accru pour les architectures autres qu'Intel

Support de la reconnaissance automatique du matériel (PnP)

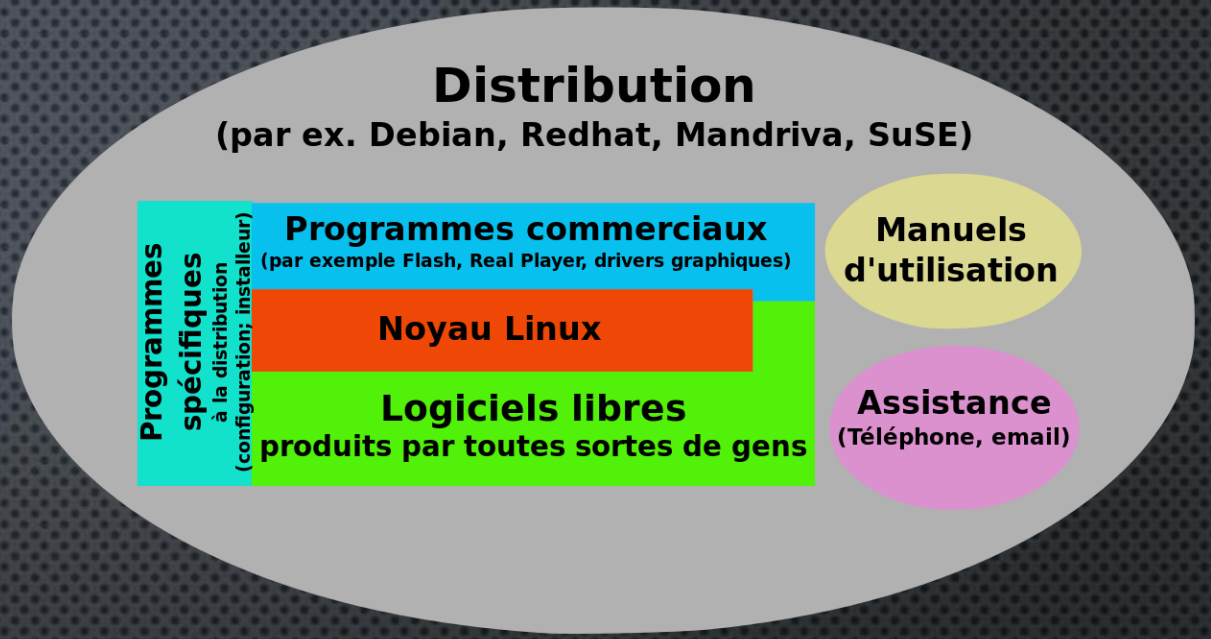
Un support et un développement communautaire (Open Source)

OPEN SOURCE

- Les êtres humains conçoivent des applications, des systèmes et des idées en langue intelligible pour les machines : du code à exécuter.
- Le terme « Open Source » correspond à l'idée que vous avez accès au code source et que vous pouvez modifier ce code.

DISTRIBUTIONS LINUX

DISTRIBUTION LINUX

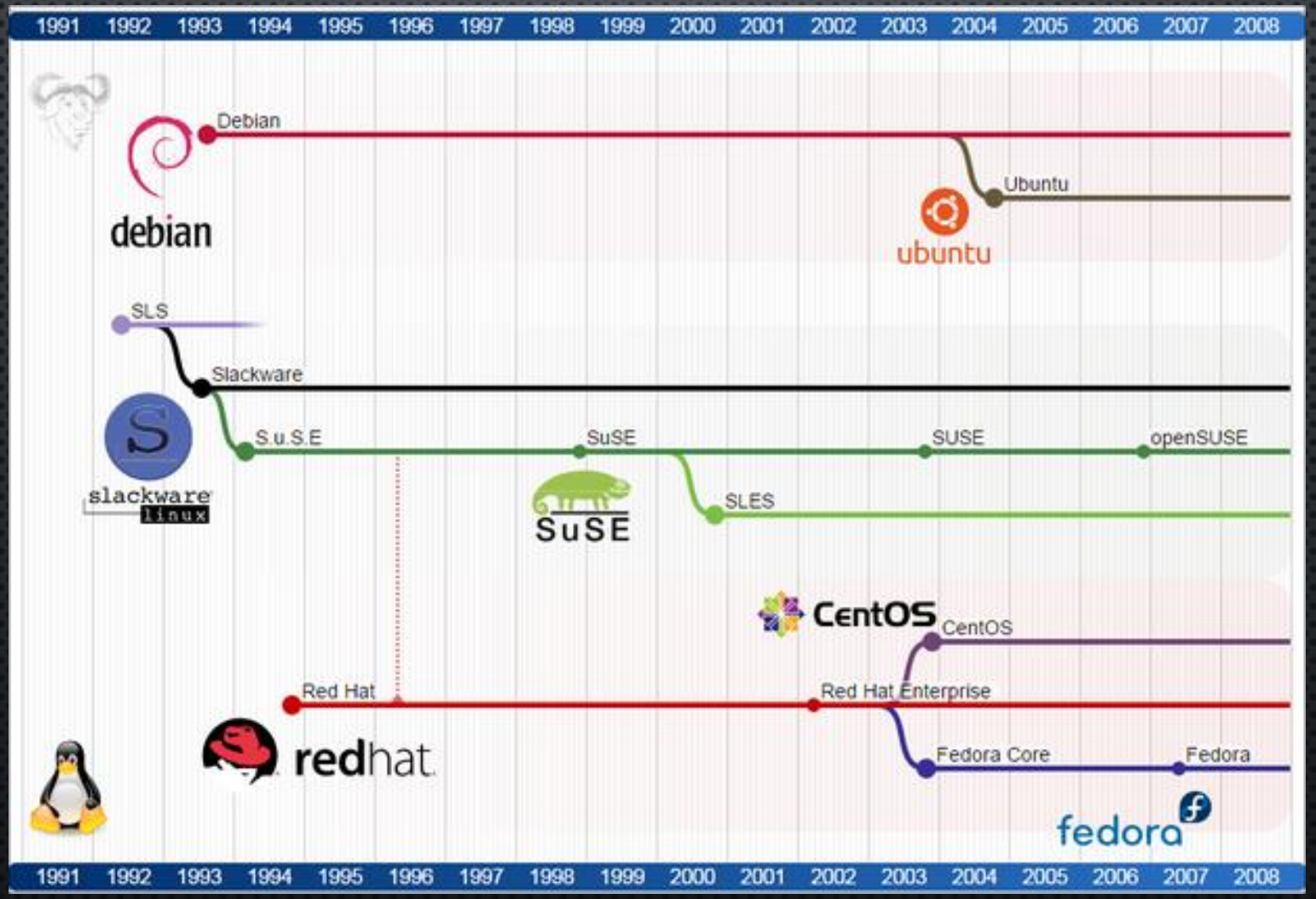


- Une distribution Linux est composée :
  - Du noyau
  - Des outils d'environnement
  - D'un logiciel d'installation
  - D'un outil de gestion des paquetages logiciels

CRITÈRES DE CHOIX D'UNE DISTRIBUTION LINUX

- Architecture matérielle : i386, amd64, arm
- Système de paquetage : dpkg, rpm, autres (opkg, pacman, emerge)
- Droits : FSF (Free Software Foundation), commercial, GNU GPL
- Stabilité : cycles de maintenance, support, End of Life (EOL)
- Usage : bureautique, mobile, serveur
- Commodité : Pratique, habitude, procédure
- Support : commercial, technique, solution SaaS

DISTRIBUTIONS GÉNÉRALISTES



DISTRIBUTIONS SPÉCIFIQUES

- Certaines distributions sont plus spécifiques. Elles disposent chacune de leur propre communauté, histoire et objectif. Par exemple,
  - OpenWRT
  - Archlinux
  - Gentoo
  - CoreOS
- Enfin, il y a aussi bon nombre de distributions spécialisées qui remplissent un objectif assez précis. Elles se basent sur l'une ou l'autre des distributions généralistes ou spécialisées.
- Kali Linux est un bon exemple : basée Debian, elle propose ses propres dépôts pour des logiciels de sécurité.
- *pfSense est une distribution spécialisée (routage et sécurité), mais elle est basée sur FreeBSD, et non pas Linux.*

CYCLE DE RÉVISION

- Un cycle de révision fournit des mises à jour et des nouvelles versions. On peut connaître des :
  - révisions mineures : corrections de bugs ou des ajouts de fonctionnalités secondaires
  - révisions majeures : nouvelles fonctionnalités, voire nouvelle conception
- Exemples :
  - Debian connait un cycle de plusieurs années
  - Ubuntu connait un cycle de 6 mois
  - Fedora est révisé tous les 6 mois
  - Une révision mineure est proposée tous les 12/18 mois chez RHEL
  - Une révision majeure est proposée tous les 3/6 ans chez RHEL

CYCLE DE MAINTENANCE

- Un cycle de maintenance est la durée pendant laquelle un logiciel est corrigé et maintenu sur un système de manière cohérente.
- Un statut EOL (End of Life) indique la fin de ce support.
- Une mise à niveau (upgrade) est nécessaire pour continuer à bénéficier d'un support de maintenance.

DEBIAN

- Distribution non-commerciale : GNU/Linux par excellence
- Support d'un grand nombre d'architectures dont ARM
- Paquetages compilés sont disponibles en dépôts locaux ou distants
  - **dpkg** : gestionnaire de paquets Debian (bas niveau)
  - **apt** : gestion avancée des paquets (gestion des dépendances, des dépôts)

PRÉSENTATION DU PROJET DEBIAN

- Debian est une organisation composée uniquement de bénévoles, dont le but est de développer le logiciel libre et de promouvoir les idéaux de la communauté du logiciel libre.
- Le projet Debian a démarré en 1993, quand Ian Murdock invita tous les développeurs de logiciels à participer à la création d'une distribution logicielle, complète et cohérente, basée sur le nouveau noyau Linux.

VERSIONS (BRANCHES) DEBIAN

- **Au 31/12/2022 :**
- Old stable (LTS) : Buster (10) jusqu'à 06/2024
- **Stable : Bullseye (11) depuis 08/2021**
- Testing : Bookworm
- Unstable : Sid (nom permanent)
- Source : https://www.debian.org/releases/

UBUNTU

- Commandité par la société Canonical et une marque déposée par cette même société.
- Basé Debian en 2004, Ubuntu respecte les licences GNU et l'esprit Open Source.
- Vise à être disponible pour tout écosystème (les télévisions, les smartphones, et les tablettes).
- Mais l'enjeu porte aussi sur le développement des technologies « cloud », notamment par un soutien fort apporté au projet Openstack.
- En savoir plus : https://doc.ubuntu-fr.org/ubuntu_distribution

VERSIONS STABLES UBUNTU

Une version standard sort 2 fois par an (supportée pendant 9 mois)

Une version LTS (Long Term Support) une fois tous les 2 ans supportée 5 ans :

2 ans pour les màj de sécurité et de pilotes matériel

3 ans en plus pour les màj de sécurité seulement

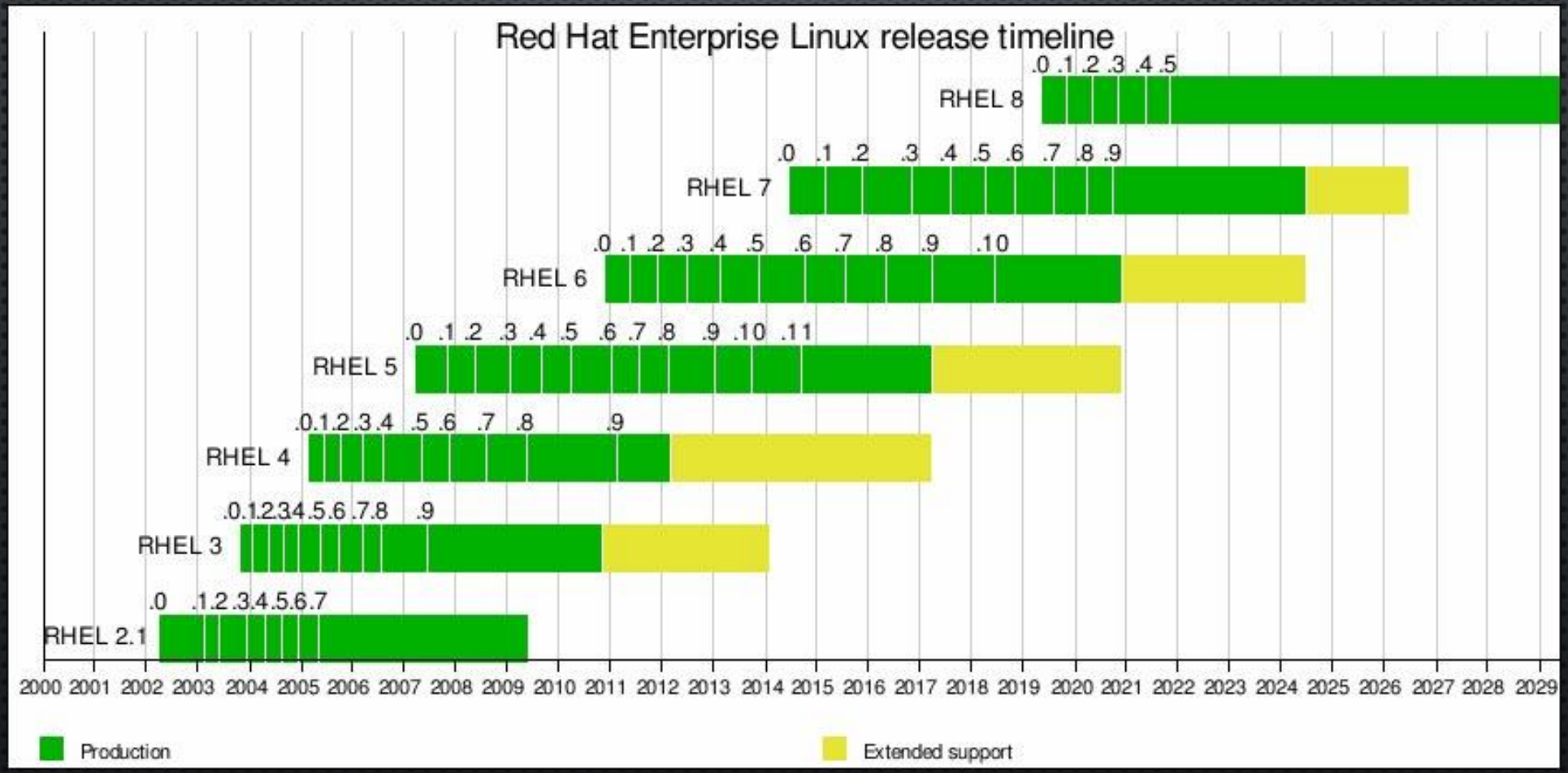
- https://fr.wikipedia.org/wiki/Liste_des_versions_d%27Ubuntu

VERSIONS UBUNTU

RED HAT RHEL

- Red Hat est une société multinationale d'origine américaine fondée en 1993 éditant des distributions GNU/Linux.
- Elle est l'une des entreprises dédiées aux logiciels Open Source les plus importantes et les plus reconnues.
- Elle distribue un OS : Red Hat Enterprise Linux (RHEL), un système d'exploitation destiné aux entreprises.
- Red Hat fournit des plateformes logicielles (openstack, JBoss), vend des abonnements d'assistance, de formations et de services d'intégration personnalisés pour les clients utilisant des logiciels open source.
- Toutes les distributions basées Redhat (CentOs, Fedora, …) utilisent le même système de paquetage **RPM**.

CYCLE DE VIE RHEL



FEDORA

- La distribution phare de Red Hat (RHEL) n'étant pas livrée gratuitement dans son format binaire, contrairement à la plupart des distributions Linux, Fedora a été créée par Red Hat pour être une distribution GNU/Linux communautaire.
- La communauté qui gère ce projet est constituée à la fois d'employés de Red Hat et de contributeurs extérieurs.
- Red Hat présente le projet Fedora comme un laboratoire pour développer de nouvelles fonctionnalités qui sont plus tard incluses dans la distribution commerciale de Red Hat.

CENTOS

- CentOS (Community enterprise Operating System) est une distribution GNU/Linux principalement destinée aux serveurs.
- ~~Tous ses paquets, à l'exception du logo, sont des paquets compilés à partir des sources de la distribution RHEL (Red Hat Enterprise Linux), éditée par la société Red Hat.~~
- ~~Elle est donc quasiment identique à celle-ci et se veut 100 % compatible d'un point de vue binaire.~~
- A compter de 2021, centOS 8 devient centOS Stream, en mode « rolling release », à l'instar de Fedora.

RED HAT PACKAGE MANAGER (RPM)

- RPM Package Manager (Red Hat Package Manager) est le logiciel de gestion des paquetages utilisé par les distributions Linux :
  - Red Hat Enterprise Linux,
  - Fedora, CentOS,
  - Mandriva,
  - openSUSE,
  - SUSE Linux Enterprise

AUTRES DISTRIBUTIONS POPULAIRES

- Archlinux
- Gentoo
- OpenWrt
- Android

- Mais aussi :
  - Kali Linux, Parrot OS
  - Damn small Linux
  - ...

ARCHLINUX

- Archlinux est une distribution légère et facile à maintenir.
- Elle dispose de dépôts pour les architectures Intel et ARM.
- Archlinux utilise le gestionnaire de paquets pacman.
- Le système de mise à jour est continu (rolling release).
- Le système de paquets est basé ABS.
- Archlinux est bien documentée : https://wiki.archlinux.fr/Accueil

GENTOO

- Gentoo Linux est une distribution dite source
- Sa particularité est la compilation complète ou partielle d'un système GNU/Linux à partir des sources, à la manière de Linux From Scratch mais automatisée.
- Ceci est géré grâce au logiciel Portage et la commande emerge en rolling release.
- C'est une distribution qui a pour objectif la portabilité.
- Gentoo est aussi très bien documentée : https://www.gentoo.org/doc/fr/

OPENWRT

OpenWrt
Wireless Freedom

- OpenWrt est une distribution GNU/Linux minimaliste pour matériel embarqué tel que des routeurs grand public basés sur des System-on-Chip Broadcom (par exemple les routeurs WLAN Belkin, TP-Link, Linksys,…) mais il est porté sur d'autres architectures.
- On compile soi-même en firmware ou une version compilée à partir d'un dépôt du projet correspondante au matériel.
- OpenWrt est capable de tenir sur une mémoire Flash de 4 Mo.
- Le gestionnaire de paquets est opkg.

DISTRIBUTIONS SPÉCIALISÉES

- On trouve depuis longtemps des distributions spécialisées qui offrent des services spécifiques déjà pré-installés.
- Ces logiciels se téléchargent librement sous format ISO, OVA / OVF ou autres (images ou recettes) et s'installent aussi bien sur des PCs, des appliances, du matériel embarqué, sur un hyperviseur ou dans le cloud.
- **Distrowatch.com** permet de faire une recherche parmi plus de 300 projets :
  - Infrastructure
  - Sécurité
  - Téléphonie
  - Pare-feu
  - Virtualisation
  - Clustering
  - Stockage (SAN)

ANDROID

- Android est défini comme étant une pile de logiciels, c'est-à-dire un ensemble de logiciels destinés à fournir une solution clé en main pour les appareils mobiles – smartphones et tablettes tactiles.
- Cette pile est organisée en cinq couches distinctes :
  - le noyau Linux avec les pilotes ;
  - des bibliothèques logicielles telles que WebKit, OpenGL, SQLite ou FreeType ;
  - une machine virtuelle et des bibliothèques permettant d'exécuter des programmes prévus pour la plate-forme Java ;
  - un framework - kit de développement d'applications ;
  - un lot d'applications standard parmi lesquelles il y a un environnement de bureau, un carnet d'adresses, un navigateur web et un téléphone.

LINUX ET WINDOWS

- Jim Zemlin, le patron de la fondation Linux, écrivait sur Twitter suite à l'annonce de Microsoft de rejoindre l'Open Innovation Network et de partager quelque 60000 de ses brevets avec la communauté open source :
  - *« Nous avons été ravis d'accueillir Microsoft en tant que membre Platinium de la Linux Foundation en 2016 et nous sommes ravis de voir leur évolution en tant que supporter à part entière de l'écosystème Linux et de la communauté open source »*
- En 2018, il existe plus de VM Linux dans Azure que sur les autres plateformes et Microsoft contribue à tous les projets open source.

LICENCES OPEN SOURCE

LICENCES LOGICIELLES

- Les droits sur un logiciel (copyright) appartiennent à son propriétaire.
- Celui-ci énonce dans une licence ce qui est autorisé avec son code.
- Il existe des licences plus permissives que d'autres.
- Le moyen le plus simple de mettre une œuvre en licence libre est de la mettre dans le domaine public

FREE SOFTWARE FOUNDATION & GPL

- La Free Software Foundation (FSF) a été créée par Richard Stallman en 1985.
- Également à l'origine des outils d'environnement GNU pour Linux et d'autres.
- GNU General Public License (GNU GPL, ou « GPL »), est une licence qui fixe les conditions légales de distribution d'un logiciel libre du projet GNU
- Introduit notamment le « copyleft » qui exige le partage de toute modification.
- Largement adopté dans le monde « libre », au-delà de GNU

FREE SOFTWARE FOUNDATION

- Aujourd'hui, un logiciel est considéré comme libre, au sens de la Free Software Foundation, s'il confère à son utilisateur quatre libertés (numérotées de 0 à 3) :
  - **(0) la liberté d'exécuter le programme, pour tous les usages**
  - **(1) la liberté d'étudier le fonctionnement du programme et de l'adapter à ses besoins**
  - **(2) la liberté de redistribuer des copies du programme (ce qui implique la possibilité aussi bien de donner que de vendre des copies)**
  - **(3) la liberté d'améliorer le programme et de distribuer ces améliorations au public, pour en faire profiter toute la communauté.**
- L'accès au code source est une condition d'exercice des libertés 1 et 3.

FREE SOFTWARE
FOUNDATION

CREATIVE COMMONS

- Licences sur des œuvres écrites, web, multimédia.
- Outil en ligne : http://creativecommons.org/choose/
- Droits de base :
  - **Attribution (BY) : citation de l'auteur**
  - **Share Alike (SA) : partage dans les même conditions**
  - **No-Derivs (ND) : on ne peut pas changer le contenu**
  - **Non Commercial (NC) : usage commercial non autorisé**
- Combinaisons comme par exemple BY-SA ou BY-NC-SA

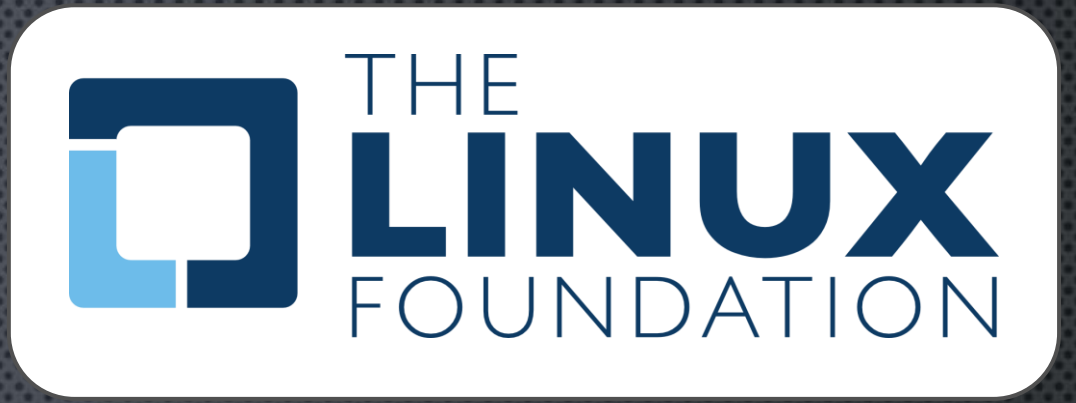
LICENCE APACHE

- Les caractéristiques majeures de la licence Apache sont :
  - Autoriser la modification et la distribution du code sous toute forme (libre ou propriétaire, gratuit ou commercial)
  - Obliger le maintien du copyright lors de toute modification (et également du texte de la licence elle-même).
  - Ce n'est pas une licence copyleft.
- Par exemple, Apache a été réutilisé comme base pour le développement d'un greffon du serveur applicatif WebSphere de chez IBM.

LICENCE MOZILLA

- La Mozilla Public License (MPL) est une licence libre créée par Netscape lors de la libération du code source de ce qui devait devenir Netscape Communicator 5 (1998).
- Celui-ci formera la base du projet Mozilla, qui utilise toujours la MPL aujourd'hui dans sa version 2.0.
- La plus grande partie du code source de Mozilla est en outre publiée sous une triple licence MPL/GPL/LGPL, ce qui permet théoriquement d'en utiliser une partie dans un logiciel GPL ou LGPL uniquement.

LINUX FOUNDATION



- La Fondation Linux est un consortium à but non lucratif fondé le 21 janvier 2007, il résulte de la fusion entre l'Open Source Development Labs et le Free Standards Group.
- La Linux Foundation a pour mission de protéger, standardiser et promouvoir Linux en procurant les ressources et services centralisés nécessaires à concurrencer de manière efficace les autres systèmes d'exploitation.

LINUX FOUNDATION

- La Linux Foundation regroupe 70 membres parmi lesquels on trouve Fujitsu, Hitachi, HP, IBM, Intel, AMD, NEC, Novell, Oracle, LG Group, Yahoo!, Samsung, Twitter et d'autres.
- On peut y adhérer individuellement pour 99$ ou gratuitement en tant qu'étudiant.
- Ces dernières années la Linux Foundation a étendu ses services à de l'organisation d'événements, de la formation et des certifications ainsi qu'au soutien de projets collaboratifs.
- Des exemples de ces projets collaboratifs sont : OpenDaylight, Open Platform for NFV (OPNFV), AllSeen Alliance, Cloud Foundry, Node.js Foundation.
- Ses domaines d'activité sont : Automotive Grade Linux, le site Linux.com, Linux Videos, Linux Developer Network, la formation, Linux Standard Base (LSB), Carrier Grade Linux, OpenPrintinget Patent Commons Project.

APPLICATIONS OPEN SOURCE

SERVICES D'ENTREPRISE

SERVEURS WEB

- La plupart des services Web sont assurés par des logiciels libres :
  - Apache,
  - Nginx,
  - LightHttpd.
- Notons aussi les accélérateurs Web :
  - Squid,
  - Varnish

BASE DE DONNÉES

- Parmi les bases de données les plus connues :
  - PostGreSQL
  - MariaDB (fork compatible et reconnu de MySQL).
- Elles permettent d'agencer des données structurées.
- Reposent sur SQL, langage d'interrogation de base de données.
- Pour d'autres approches (noSQL) ou d'autres usages :
  - MongoDB,
  - REDIS,
  - SQLite,
  - Zope Database, ...

SERVEURS MAIL **MTA** (MAIL TRANSPORT AGENT)

- Un service MTA (SMTP) transfère le courrier électronique à travers l'Internet. On citera :
  - Sendmail,
  - Postfix,
  - Exim, ...

MDA/MUA OPEN SOURCE

- MDA : Mail Delivery Agent
- Un service MDA (POP3/IMAP) livre le courrier électronique aux utilisateurs. On citera :
  - Cyrus,
  - Dovecot.
- MUA : Mail User Agent
- Des logiciels MUA comme procmail ou beaucoup d'autres permettent de récupérer le courrier. On peut le faire également via des interfaces Web :
  - SquirrelMail,
  - Roundcube,
  - Horde, ...

SERVEURS DE FICHIERS OPEN SOURCE

- Linux offre des services de fichiers pour une panoplie de protocoles :
  - FTP : proFTPD, Vsftpd
  - NFS : support natif
  - CIFS/SMB : Samba client et server qui pourra jusqu'à reproduire à 90% un serveur Active Directory
- SSH dispose de deux sous-protocoles qui permettent avantageusement de transmettre des fichiers de manière sécurisée : SCP et SFTP.
- Rsync permet de maintenir une synchronisation des copies.

SERVICES D'INFRASTRUCTURE

- **ISC-DHCP** permet de gérer un réseau en offrant un service robuste DHCP et DHCPv6
- **ISC-Bind** offre un service robuste DNS, certainement le plus utilisé dans le monde.
- **OpenLDAP** offre un service d'annuaire LDAP réputé.
- **Samba4** permet de reproduire un environnement Active Directory. Il est largement mis en production et Microsoft collabore dans une certaine mesure.
- **NFS**

SERVICES COLLABORATIFS

- BlueMind
- NextCloud
- eGroupware
- Zimbra
- Open-Xchange
- RoundCube

SERVICES DE TÉLÉPHONIE

- Asterisk (IPBX)
- Kamailio (Serveur SIP)
- Freeswitch
- FreePBX
- Ast2Billing
- XiVo

CLOUD COMPUTING

- OpenNebula,
- OpenStack,
- Eucalyptus,
- Cloud Stack,
- Nimbus

VIRTUALISATION

- Qemu,
- KVM,
- OpenVZ,
- Xen,
- Virtualbox,
- Proxmox,
- O-virt
- Jail,
- LXC,
- Docker

GESTION DE PARC

- GLPI,
- OCS Inventory NG,
- Fusion Inventory

POSTE À DISTANCE

- Tight VNC,
- SSH,
- X2Go

AUTOMATION, ORCHESTRATION

- Puppet,
- Chef,
- Vagrant,
- Ansible
- Kubernetes

SAUVEGARDE

- Bacula,
- Partimage,
- Amanda,
- CloneZilla

HAUTE DISPONIBILITÉ

- HAProxy,
- Keepalived,
- Linux-HA (heartBeat)

SÉCURITÉ

- AIDE,
- OpenVas,
- ClamAV,
- Snort,
- Wireshark,

VPN

- OpenSwan,
- OpenVPN

FIREWALL

- NetFilter,
- Packet Filter,
- pfSense,
- NuFW,
- Firewalld

SURVEILLANCE

- Nagios,
- Cacti,
- Centreon,
- MRTG,
- Munin,
- OpenNMS,
- Zabbix,
- Zenoss,
- Icinga,
- Shinken

PKI (PUBLIC KEY INFRASTRUCTURE)

- EasyCA,
- OpenCA PKI,
- EJBCA,
- OpenSSL

FENÊTRES GRAPHIQUES OPEN SOURCE

- Un service de fenêtres graphiques permet de les ouvrir, de les redimensionner, etc…
- Une distribution « desktop » vient d'emblée avec tous les outils graphiques.
- **X-Window** est la base du système graphique, il fournit les fenêtres et les primitives de base.
- Compiz, FVWM, Enlightenment, Metacity sont des gestionnaires de fenêtres.

ENVIRONNEMENTS DE BUREAU

- Les environnements de bureau offrent un service complet de fenêtrage et de l'interface graphique avec l'ordinateur.
- Au même titre que les services X, il peut être déporté à distance. Il peut même être chiffré via SSH.
- On citera :
  - Unity,
  - Gnome Shell,
  - KDE,
  - Mate,
  - Xfce,
  - LXDE, et bien d'autres ...

SUITE DE PRODUCTIVITÉ / BUREAUTIQUE

- Suite bureautique : **LibreOffice** (fork d'OpenOffice)
- On citera **Iceweasel** (version GNU de Mozilla Firefox) comme navigateur Web, ou Chromium
- Thunderbird, Evolution et KMail sont des clients mail collaboratifs célèbres.
- Création graphique : Gimp, Inkscape, Scribus
- CAO : LibreCAD, FreeCAD,
- Vidéo : KdenLive, Natron

ENVIRONNEMENTS DE DÉVELOPPEMENT

- Langage de programmation : C, C++, Java, Perl, Python, PHP, Ruby
- Plateforme de développement : Redmine, GIT, Eclipse, CVS, Subversion
- Plateforme de développement Web : Django, JQuery, Ruby On Rails, Zend Framework, Node.js, REDnode

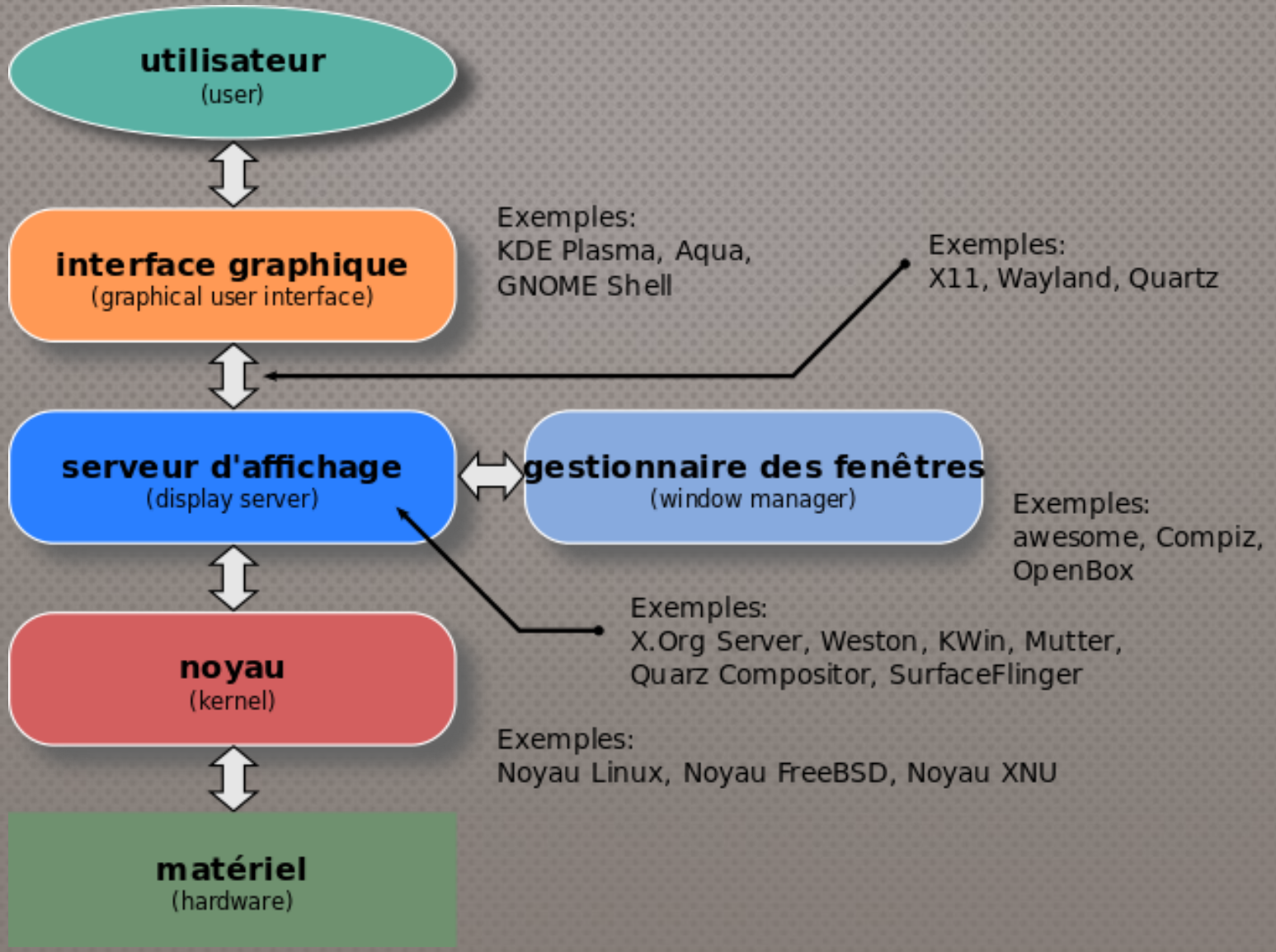
APPLICATIONS CMS, E-COMMERCE, ERP

- CMS et blogs : Drupal, Wordpress, Joomla, Spip, Plone, Ghost
- E-Commerce : Magento, Prestashop, Oscommerce
- ERP : Compiere, Dolibarr, Odoo

ENVIRONNEMENTS DE BUREAU

ENVIRONNEMENTS DE BUREAU

- Un environnement de bureau est un ensemble de programmes qui permettent de manipuler l'ordinateur à travers une interface graphique qui fait analogie à un bureau.
- De nombreux systèmes d'exploitation ont un environnement de bureau incorporé.
- À l'inverse, avec le système de fenêtrage X des systèmes d'exploitation Unix, plusieurs environnements de bureau sont disponibles.
- C'est ainsi qu'un environnement de bureau Linux est composé de plusieurs éléments distincts :
  - le système de fenêtrage (X11, X Window System, Mir)
  - le gestionnaire de fenêtres (Compiz, KWin, Metacity, Xfwm, ...)
  - l'environnement graphique
  - le gestionnaire de fichiers et divers outils
  - le gestionnaire de d'affichage, Display Manager qui ouvre les sessions (GDM, LightGDM, KDM, XDM)


utilisateur
(user)
interface graphique
(graphical user interface)
Exemples:
KDE Plasma, Aqua,
GNOME Shell
Exemples:
X11, Wayland, Quartz
serveur d'affichage
(display server)
gestionnaire des fenêtres
(window manager)
Exemples:
awesome, Compiz,
OpenBox
Exemples:
X.Org Server, Weston, KWin, Mutter,
Quarz Compositor, SurfaceFlinger
noyau
(kernel)
Exemples:
Noyau Linux, Noyau FreeBSD, Noyau XNU
matériel
(hardware)

ENVIRONNEMENTS DE BUREAU LINUX

- Environnements traditionnels : GNOME3, KDE4
- Environnements dérivés : Unity (GNOME3, défaut Ubuntu), Cinnamon (GNOME3, défaut Linux Mint), Mate (GNOME)
- Environnements légers : LXDE, Xfce
- Environnement entièrement paramétrable : Enlightenment, Environnement Qt, Razor-qt, Elokab

#1

LE SHELL

L'interpréteur de Commandes GNU/Linux

LE SHELL – LA LIGNE DE COMMANDE

- La **ligne de commande (CLI)** est un moyen simple d'interagir avec un ordinateur.
- Le shell **interprète** les commandes tapées au clavier.
- Le shell ne fait partie intégrante du noyau (kernel), c'est un programme comme un autre ; il est d'ailleurs possible d'en changer.
- Le **prompt**, ou « invite de commande », qui se termine par un $ pour un utilisateur standard ou un # pour l'administrateur du système (root), indique que le shell attend les commandes de l'utilisateur.
- Le shell est également un langage de programmation que l'on peut utiliser pour lancer des tâches automatiquement (au travers de **scripts**)

LE SHELL – DÉFINITIONS

- Le terminal = l'environnement d'entrée/sortie
- La console = terminal physique (teletype = tty)
- Shell =
  - Interpréteur de commande (« lancer » des commandes)
  - Environnement : confort de l'utilisateur, sécurité
  - Langage de programmation
  - Traitement du texte (éditeur et autres outils)
  - Interface avec le noyau

LE SHELL - TYPES DE SHELLS

- On obtient la liste des shells présents sur le système en affichant le fichier /etc/shells :
- `$ cat /etc/shells`
  - sh : historique, standard, portable (Bourne Shell)
  - csh/tcsh : C Shell
  - ksh : Korn Shell
  - …
  - bash : Bourn Again Shell GNU/Linux, le plus utilisé

BASH : BOURNE AGAIN SHELL

- Le projet GNU offre des outils pour l'administration de système de type UNIX qui sont libres et qui respectent les standards UNIX.
- **Bash** est un Shell compatible avec sh (Bourne Shell) qui incorpore des spécificités utiles du Korn Shell (ksh) et du C Shell (csh).
- Il est censé se conformer à la norme **POSIX** (IEEE POSIX P1003.2/ISO 9945.2 Standards des Shell et Outils.
- Il offre des améliorations fonctionnelles par rapport à sh pour la programmation et l'utilisation interactive.

LE SHELL INTERACTIF

- Quand on obtient un terminal avec une ligne de commande, on si situe dans un environnement encadré par un programme « shell » qui a créé un **processus** sur le système.
- Il permet notamment d'exécuter des commandes.
- Par exemple, La commande echo permet d'afficher du texte à l'écran :

```
root@debianA:~#
root@debianA:~# echo affiche du texte
affiche du texte
root@debianA:~# echo "avec ou sans guillemets"
avec ou sans guillemets
root@debianA:~#
```

LE SHELL INTERACTIF

```
root@debianA:~#
root@debianA:~# echo affiche du texte
affiche du texte
root@debianA:~# echo "avec ou sans guillemets"
avec ou sans guillemets
root@debianA:~#
```

- On remarque le prompt composé de :
  - **root** = l'utilisateur connecté
  - **@** = séparateur (user@machine)
  - **debianA** = nom de l'ordinateur (hostname)
  - **~** (tilde) = le dossier courant est le dossier de base de l'utilisateur (home)
  - **#** = indique le type de connexion, ici un superutilisateur (il a tous les pouvoirs) ; un **$** ou **>** indique une connexion standard (utilisateur sans pouvoir particulier)
- *Cette configuration de l'environnement est chargée sous forme de script au moment de la connexion de l'utilisateur (variable d'environnement PS1)*

```
pascal@debianA:~$ echo "je n'ai pas trop de pouvoir"
je n'ai pas trop de pouvoir
pascal@debianA:~$ _
```

UTILISER LE SHELL

- Dans la suite, l'invite de commande sera représentée simplement par un **#** ou un **$**, selon le contexte souhaité (root ou pas root).
- Au sein du terminal, le clavier s'utilise de façon standard
- On se déplace avec les flèches gauche et droite, on supprime avec [Retour arrière] ou [Suppr], et on valide la saisie avec [Entrée]
- D'autres raccourcis peuvent être utiles :
  - [ctrl] + A : aller au début de la ligne
  - [ctrl] + E : aller à la fin de la ligne
  - [ctrl] + L : efface le contenu du terminal
  - [ctrl] + U : effacer la ligne jusqu'au début
  - [ctrl] + K : effacer la ligne jusqu'à la fin
- Essayez quelques commandes basiques : **date**, **cal**, **pwd**, **ls**

UTILISER LE SHELL

- Pour entrer des commandes dans le shell, il faut :
  - une commande valide (reconnue par le système)
  - suivie éventuellement d'une ou plusieurs options notées
    - par un « dash », le tiret « - » en notation abrégée
    - ou un double « dash », double tiret « -- » en notation extensive,
  - des arguments,
  - et un retour chariot qui soumet la ligne entrée.

SHELL - SYNTAXE DES COMMANDES

- Chaque commande dispose de sa propre syntaxe :
- sans options :
  - **`$ ls`**
- avec une option :
  - **`$ ls -l`**
- avec plusieurs options :
  - **`$ ls -l -a -h –t`**
  - *Que l'on peut condenser :*
  - **`$ ls -laht`**

SHELL - OPTIONS ET ARGUMENTS

- Options double dash (notation extensive) :
  - **`$ ls --all`**
  - **`$ ls --help`**
- Donner un argument :
  - **`$ ls -l /home`**
- Donner plusieurs arguments :
  - **`$ ls -l /home /var`**

LE « PATH »

- Par exemple, la commande ls :
  - `$ ls`
- s'exécute car elle est située dans un des chemins indiqués dans le **PATH**.
- Le **PATH** est une variable d'environnement qui contient la liste des répertoires « visités » par le Shell pour trouver la commande qui a été saisie (le fichier binaire exécutable associé).
- Le binaire **ls** se situe par exemple dans le répertoire **/usr/bin**

LE « PATH »

- On obtient le contenu de la variable PATH avec :
  - **`$ echo $PATH`**
  - `/usr/local/sbin:/usr/local/bin:/usr/sbin:/`**`usr/bin`**`:/root/bin`
- On peut exécuter la commande en indiquant son emplacement absolu :
  - **`$ /usr/bin/ls`**
- Ou encore :
  - **`$ cd /usr/bin`**
  - **`$ ./ls -l ls`**
- Dans ce cas, on ne sollicite pas le PATH

SHELL – ENTRER DES COMMANDES

- On peut entrer des commandes sur plusieurs lignes :
  - `$ ls /var`
  - `$ ls /home`
  - `$ ls /usr`
- Ou sur une seule ligne ; on sépare les commandes par un point-virgule :
  - `$ ls /var; ls /home; ls /usr`

SÉQUENCES DE COMMANDES

- Si on veut réaliser une commande avec différents arguments, on peut créer une boucle et profiter de variables :
  - **$ for arg in /home /var /usr**
  - **> do**
  - **> echo "visualisation de : " $arg**
  - **> ls -a $arg**
  - **> done**
- ou encore en une seule ligne :
- **$ for arg in /home /usr /var ; do ls -a $arg ; done**

EXÉCUTIONS CONDITIONNELLES

- **&&** et || sont des séparateurs de commandes conditionnels.
- **Opérateur &&**
- $ *commande1* && commande2
- exécute commande2 si commande1 est exécutée sans erreur. Par exemple sous Debian :
  - **`# apt update && apt -y upgrade`**

EXÉCUTIONS CONDITIONNELLES

- **Opérateur** ||
- $ commande1 || commande2
- exécute commande2 si commande1 est exécuté **avec** erreur.
- Par exemple :
  - `# yum -y update || apt update`

#2

HISTORIQUE DES COMMANDES

- Pour voir la liste des commandes que vous avez validées, vous pouvez utiliser la commande interne de bash **history** :
  - `$ history`
- La commande **history** liste les commandes en cache ainsi que celles sauvées dans : **~/.bash_history**.
- Lorsque l'utilisateur quitte le shell, les commandes en cache sont inscrites dans ce fichier.
- Vous pouvez récupérer les commandes tapées en utilisant les flèches directionnelles [haut] et [bas] de votre clavier.
- **history -c** efface l'historique de la session courante.

HISTORIQUE : RACCOURCIS « BANG »

- A l'aide des exemples suivants, essayez de trouver l'utilité des caractères spéciaux **^** et **!** (bang) :
  - **`$ history -c`**
  - **`$ ls -a`**
  - **`$ ^ls^ps`**
  - **`$ !!`**
  - **`$ history`**
  - **`$ !2`**

HORODATAGE DE L'HISTORIQUE (EXERCICE)

- Modifier le fichier suivant :
  - **`$ nano ~/.bashrc`**
- Ajouter le paramètre suivant :
  - **`export HISTTIMEFORMAT='%F %T :'`**
- Sauvegarder le fichier, et appliquez les changements avec :
  - **`$ source ~/.bashrc`**

HISTORIQUE

- La commande **history** est une primitive du Shell
- Une primitive quasi-équivalente est :
  - `$ fc -l`
- Avec l'option **`-l`**, la primitive **`fc`** affiche les 15 dernières commandes de l'historique
- Pourquoi parle-t-on de ***primitive*** ?

LA COMMANDE TYPE

- Il existe deux types de commandes Shell
  - Les commandes **internes** : ces commandes font partie intégrante du code informatique du Shell lui-même. On parle de « **primitive** » du Shell.
  - Les commandes **externes** : elles correspondent à un **programme exécutable** extérieur au code du Shell, et en général présent sur le disque dur de l'ordinateur.
- La commande (primitive) **type** permet de connaître le type de commande (et son emplacement si elle est externe), par la syntaxe :
  - **`$ type commande`**
- Essayez avec pwd, cd, ls, cal (ou toute autre commande que vous connaissez).

TOUCHE TABULATION

- Selon la distribution, la touche de tabulation offre des possibilités d'auto-complétion.
- Par exemple :
  - `$ hi`
- Suivi de la touche [TAB] complète la commande (s'il n'y a pas d'ambiguïté)

SUBSTITUTION DE COMMANDES

- Il s'agit de récupérer la sortie d'une commande dans une variable.
- Par exemple : la commande **uname** permet de connaître la version du noyau courant. Comment la substituer ?
  - **`$ uname -a`**
- Vous obtenez par exemple : `Linux Debian 4.9.0-8-amd64 #1 …`
- Maintenant, récupérons cette information dans la variable system :
  - **`$ system=$(uname -a)`**
- Puis affichons cette variable :
  - **`$ echo $system`**

ALIAS

- Un alias est une autre manière de substituer des commandes.
- Liste des alias :
  - **$ alias**

```
...
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto’
```

ALIAS

- Pour créer un alias, on utilise la même commande, avec un argument
- Essayez par exemple ceci :
  - `$ alias zozo='ls -a'`
  - `$ alias h="history"`
  - `$ alias`
  - `$ zozo`
  - `$ h`
- *NB : les alias créés ne resteront actifs que le temps de la session. Pour les rendre permanents, il faut les intégrer dans le script* `~/.bashrc`

MÉTA-CARACTÈRES

- Les méta-caractères ont un sens spécial pour le shell.
- Ils sont la plupart du temps utilisés comme jokers, pour correspondre à plusieurs noms de fichiers ou de dossiers en utilisant un minimum de lettres.
- Les caractères d'entrée (<), de sortie (>) et le tube (|) sont également des caractères spéciaux ainsi que le dollar ($) utilisé pour les variables.
- Notez que ces caractères sont rarement utilisés pour nommer des fichiers standards (ce n'est pas du tout recommandé).
- *NB : le caractère bang (!) est d'un genre encore différent, il est appelé mot-clé du shell :*
  - `$ type !`

CARACTÈRES GÉNÉRIQUES OU JOKERS

- **Masque générique ***
- Le caractère * remplace n'importe quel nombre de caractères.
- Par exemple :
  - `$ ls  /usr/bin/b*`
- Liste tous les programmes commençant par « b »
- On traduit donc « b* » par « **b** suivi de n'importe quoi »
- Y compris rien du tout.

CARACTÈRES GÉNÉRIQUES OU JOKERS

- **Masque de caractère ?**
- Le caractère ? remplace n'importe quel caractère **unique**.
- Par exemple :
  - `$ ls  /usr/bin/?b*`
- Liste tous les programmes ayant un « b » pour deuxième lettre.

CARACTÈRES GÉNÉRIQUES OU JOKERS

- **Plage de valeurs [ ]**
- [ ] est utilisé pour définir une plage de valeurs :
  - `$ ls /usr/bin/linux[0-9][0-9]`
- liste tous les fichiers commençant par « linux » suivi de deux chiffres.
  - `$ ls /usr/bin/[!Aa-Yy]*`
- liste tous les fichiers dont la première lettre n'est pas dans l'intervalle [a-y] ou [A-Y] (le ! Indique ici une négation)

CARACTÈRES GÉNÉRIQUES OU JOKERS

- **Filtre {chaîne1,chaîne2}**
- {chaîne1,chaîne2} même si ce n'est pas simplement un joker de nom de fichiers, on peut l'utiliser pour filtrer des noms de fichiers :
  - **`$ ls index.{htm,html}`**
- Essayez par exemple :
  - **`$ ls /usr/bin/*{rch,ast}`**
  - **`$ ls /usr/bin/?{rch,ast}`**

SHELL SCRIPTING

NOTRE PREMIER SCRIPT SHELL

POURQUOI ÉCRIRE UN SCRIPT SHELL ?

- Simplifier et/ou automatiser une procédure complexe
- Déploiement logiciel : installateur de binaire, avec compilation, avec dépôts de paquetages
- Supervision de parc, exploitation de journaux, tâches de maintenance périodiques
- Configuration, gestion, maintenance, surveillance d'environnements virtualisés
- Gestion spécifique à la production :
  - Bases de données
  - Services Web
  - Services d'infrastructure
  - Services de traitement (batch)
- ...

EDITEUR DE TEXTE

- Un bon éditeur de texte est indispensable.
- On pourra changer l'éditeur par défaut sous Debian avec la commande :
  - **`$ sudo update-alternatives --config editor`**
- Globalement, deux écoles :
  - **vi** : éditeur historique (1976) Unix (**vim** est la version GPL) ; rapide mais difficile
  - **nano** : notepad « like » devenu très populaire
- Il existe d'autres éditeurs : emacs, mcedit.

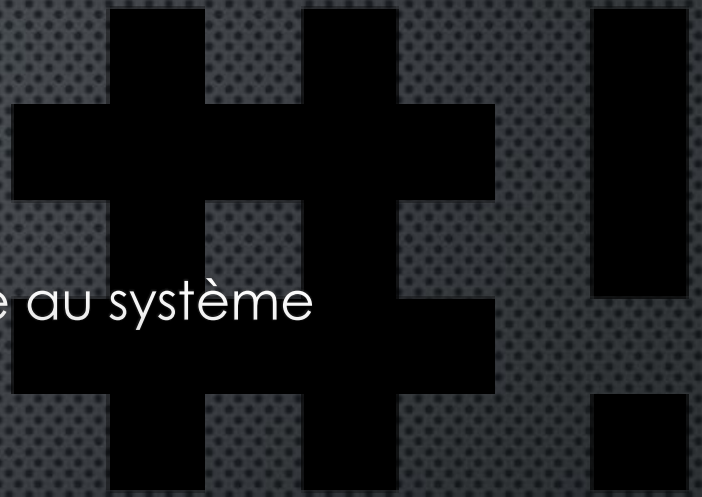
INVOCATION DE L'INTERPRÉTEUR

- Créez un premier script nommé **monscript.sh** avec nano ou vi (dans votre répertoire personnel) :
  - `echo "ceci est mon premier script"`
- Invoquez ce script monscript.sh avec sh :
  - `$ sh monscript.sh`
- Renommez **monscript.sh** en **monscript** :
  - `$ mv monscript.sh monscript`
- Ajoutez une commande à monscript :
  - `$ echo 'ceci est ma seconde ligne'`
- Invoquez ce script monscript avec sh :
  - `$ sh monscript`
- On voit que l'extension de fichier de script n'a aucune signification pour le shell

APPEL DIRECT DU SCRIPT

- Objectif : rendre le script accessible directement depuis la ligne de commande
- Il faut d'abord rendre fichier exécutable :
  - **`$ chmod u+x monscript`**
- Plusieurs méthodes :
  - Placer le script dans un chemin du **PATH**
  - Modifier la variable **PATH** en ajoutant l'endroit du fichier soit l'emplacement local courant. Avec cette méthode, tout script présent à l'endroit courant peut être appelé.
  - Appel depuis son emplacement original :
    - **`$ ./monscript`**

LE SHEBANG



- Le Shebang (contraction de Sharp + Bang = #!) indique au système l'interpréteur à utiliser pour lancer les commandes :
  - **`#!/bin/bash`**
- Ou, souvent préféré :
  - **`#!/usr/bin/env bash`**
- Si l'interpréteur de commandes n'est pas déclaré, le shell courant prend en charge les commandes du script.
- Pour déclarer l'interpréteur à utiliser pour un script donné, il faut ajouter le Shebang sur la première ligne du script.
- *NB : il peut s'agir d'un shell Unix, mais aussi d'un interpréteur d'un autre type (Python, PHP, Perl, …)*

EXERCICES (MANIPULATIONS)

- Manipuler la variable PATH :
  - **`$ PATH=`**
  - Quel est l'effet produit ? Comment y remédier ?
  - Quelle est l'utilité d'ajouter le répertoire . (point) à la fin de votre variable PATH de cette manière :
    - **`$ PATH=$PATH:.`**
- Obtenir la liste des variables d'environnement :
  - **`$ env`**
- Modifier le paramètre linguistique :
  - **`$ echo $LANG`**
  - **`$ date ; cal ; ls -l`**
  - **`$ LANG=en_US.UTF-8`**
  - **`$ date ; cal ; ls -l`**
  - **`$ LANG=fr_FR.UTF-8`**

CONFIGURATION LOCALES

CONFIGURATION DES PARAMÈTRES RÉGIONAUX (LANGUES, CLAVIERS)

DEBIAN – CONFIGURATION DES LOCALES

- Modifier le clavier :
  - **`$ sudo dpkg-reconfigure keyboard-configuration`**
- Modifier les « locales » (en France, c'est : **`fr_FR.UTF-8`**)
  - **`$ sudo dpkg-reconfigure locales`**
- On vérifie les paramètres :
  - **`$ locale`**

AIDE SOUS LINUX

Outils et Commandes de base pour obtenir de l'aide

COMMANDES LESS ET MORE

- **less** est une commande Unix permettant de visualiser un fichier texte page par page (sans le modifier).
- Elle est similaire à la commande **more**, mais permet en plus de revenir en arrière ou de rechercher une chaîne.
- Contrairement à **vi** ou **nano** (qui permettent aussi de visualiser des fichiers), **less** n'a pas besoin de charger entièrement le fichier en mémoire et s'ouvre donc très rapidement même pour consulter de gros fichiers.
- Raccourcis disponibles dans **less** :
  - **h** ou **help** pour l'aide
  - **/** suivi d'une occurrence pour effectuer une recherche
  - Barre d'espace : pour avancer d'une page
- Exemples :
  - **`$ less /var/log/dpkg.log`**

COMMANDE MAN

- **man** est une commande Unix. Elle permet de visionner le manuel d'une commande du shell
- Si la commande ne fonctionne pas, on installe le paquet :
  - `# apt install man`

APPEL D'UNE PAGE MANUEL

- Pour appeler une page de manuel, entrer tout simplement :
  - **$ man [-s<section>] <nom_de_commande>**
- Par exemple :
  - **$ man man**
  - **$ man ls**
  - **$ man 5 passwd**
  - **$ man -s 5 passwd**
  - **$ man passwd.5**

RECHERCHE D'UNE PAGE MANUEL

- Une page de manuel peut avoir le même nom et faire partie d'une section différente (la portée de la page est différente).
- Par exemple :
  - **`$ man -f passwd`**
  - **`passwd (1)`** **`- Modifier le mot de passe d'un utilisateur`**
  - **`passwd (5)`** **`- fichier des mots de passe`**
  - **`passwd (1ssl)`** **`- calcul des hachages des mots de passe`**
- L'option **man -f passwd** permet d'effectuer une recherche sur le nom des pages man.
- La commande **whatis passwd** a le même effet.

CONTENU D'UNE PAGE MANUEL

- En en-tête de la page man on trouve le nom de la commande suivie d'un numéro entre parenthèses. Il s'agit du numéro de section.
- Une page est composée de plusieurs parties, selon ce schéma (non exhaustif) :
  - NOM,
  - SYNOPSIS,
  - CONFIGURATION,
  - DESCRIPTION,
  - OPTIONS,
  - CODE DE RETOUR, VALEUR RENVOYÉE, ERREURS, ENVIRONNEMENT, FICHIERS, VERSIONS, CONFORMITÉ, NOTES, BOGUES, EXEMPLE, AUTEURS,VOIR AUSSI, TRADUCTION.

CONTENU D'UNE PAGE MANUEL

- Le **SYNOPSIS** indique brièvement l'interface de la commande ou de la fonction (syntaxe et ses arguments) :
  - Les caractères **gras** marquent le texte invariable
  - Les *italiques* indiquent les arguments remplaçables.
  - Les crochets **[ ]** encadrent les arguments optionnels,
  - les barres verticales **|** (caractère pipe) séparent les alternatives,
  - les ellipses **...** signalent les répétitions.

COMMANDES MAN -K OU APROPOS

- **`$ man -k passwd`**
- Cette commande recherche la description courte et le nom des pages de manuel comportant le mot-clé indiqué, utilisé comme une expression rationnelle, puis affiche tout ce qui a été trouvé.
- La commande apropos donne l'équivalent :
  - **`$ apropos passwd`**
- La commande whatis effectue une recherche similaire :
  - **`$ whatis -r passwd`**
- Man utilise une base donnée pour consulter les descriptions des pages. En cas de pages ou de logiciels ajoutés, il est indiqué de mettre à jour la base de données mandb:
  - **`# mandb`**

COMMANDE INFO

- info a d'abord été fourni avec le paquet Texinfo de GNU en alternative plus exhaustive et documentée que les pages de manuel UNIX et a été porté par la suite sur d'autres systèmes de type Unix.
  - `$ info`
- Info fonctionne de façon interactive, avec un concept proche de l'hyperlien.

AUTRES SOURCES D'AIDE

- Avec l'option **-h** ou **--help** d'une commande :
  - **`$ man --help`**
- La commande **info** (projet GNU) :
  - **`$ info`**
- Les fichiers README des sources,
- dans le dossier **/usr/share/doc**
- Sans oublier les innombrables manuels, wikis, cours, forums, blogs, … consacrés à Linux.

CONNAÎTRE LA VERSION DE LA DISTRIBUTION

- La commande **uname** permet d'obtenir des informations sur le système
  - **`$ uname -srv`**
- Communément le fichier **/etc/os-release** donnera des informations concernant la version de la distribution :
  - **`$ cat /etc/os-release`**
- Voir aussi :
  - **`$ cat /proc/version`**
- Et sous les distributions Debian et dérivées, on a aussi :
  - **`$ lsb_release –a`**
- Essayer aussi avec les options –as ; –cs ; –ds ; –rs
- Voir aussi :
  - **`$ cat /etc/debian_version`**

#3

Traitement du texte

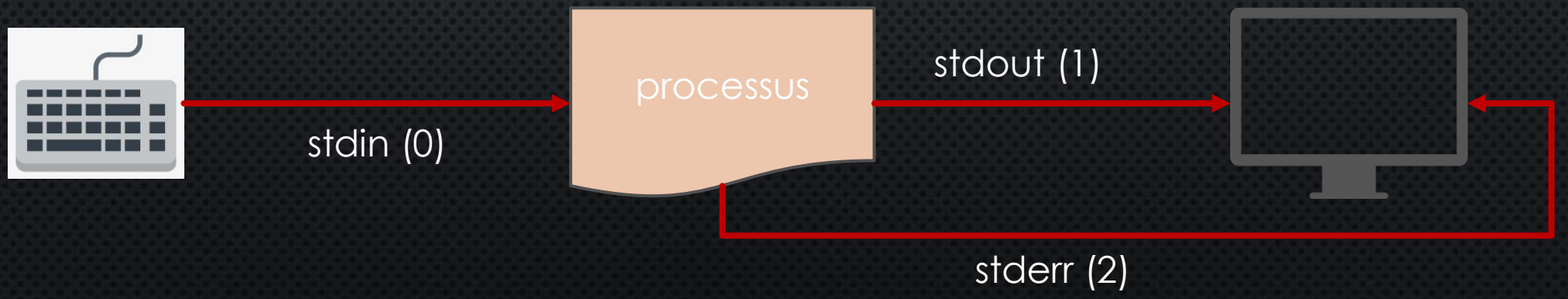
Redirection, Analyse et édition du Texte

OUTILS DE BASE

DE TRAITEMENT DU TEXTE

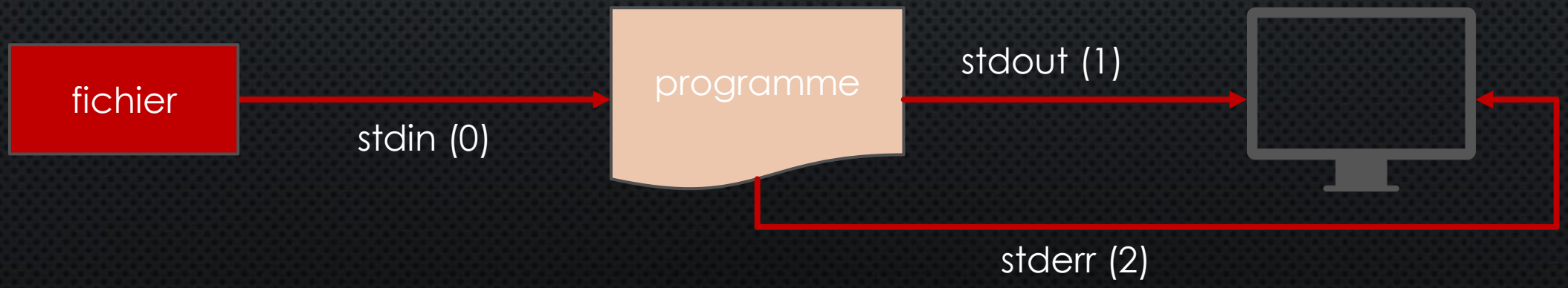
REDIRECTIONS ET TUBES

- Les processus UNIX ouvrent trois descripteurs de fichiers standards (correspondant aux flux standards) qui permettent de traiter les entrées et sorties.
- Ces descripteurs standards peuvent être redéfinis pour chaque processus
- Dans la plupart des cas, le descripteur **stdin** (entrée standard) est le clavier, et les deux descripteurs de sortie, **stdout** (sortie standard) et **stderr** (l'erreur standard), sont l'écran.



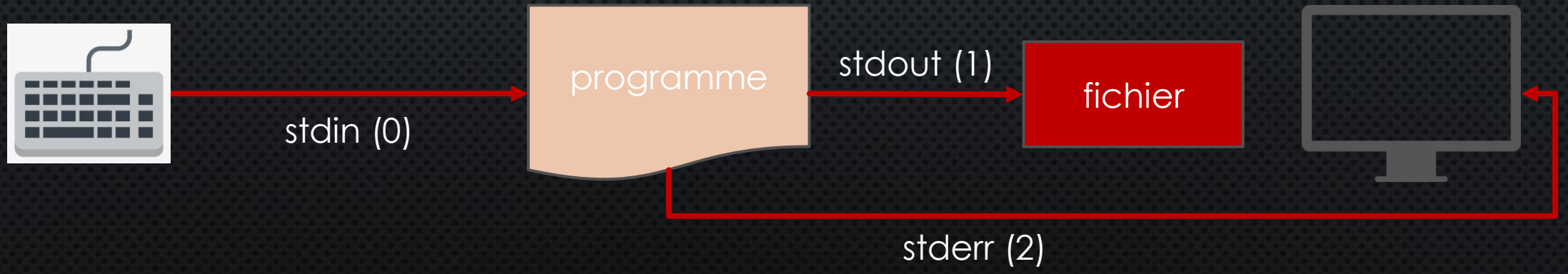
REDIRECTION DE L'ENTRÉE STANDARD

- On peut « alimenter » les commandes passées à un processus par une autre source que l'entrée standard (le clavier), comme un fichier ou un autre périphérique, avec l'opérateur < :
  - **`$ programme < fichier`**
- Dans ce cas, les données vont de droite à gauche. L'opérateur < ne peut être utilisé qu'avec **stdin** : on ne peut pas l'utiliser avec les flux de sortie.
- On peut aussi écrire (0 désignant l'entrée standard stdin) :
  - **`$ programme 0< fichier`**



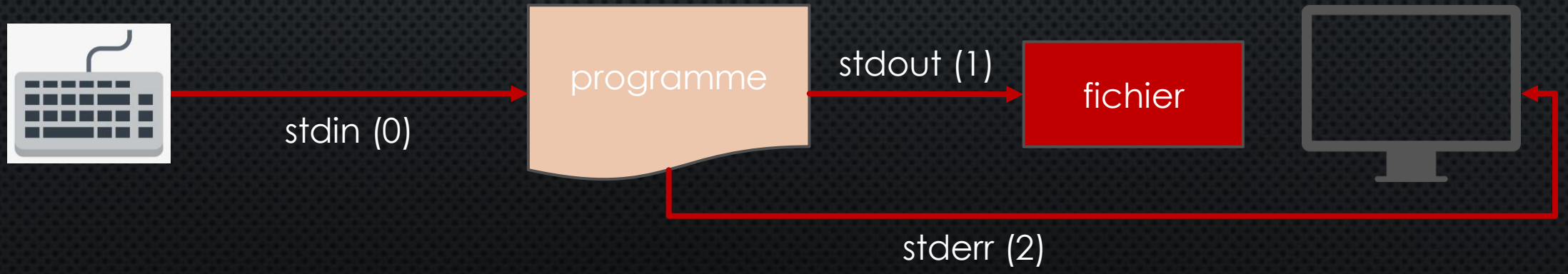
REDIRECTION DE LA SORTIE STANDARD

- Dans le même esprit on peut diriger la sortie d'un programme vers un fichier ou un périphérique, plutôt que sur la sortie standard (l'écran) :
  - **`$ programme > fichier`**
- Dans ce cas, les données vont de gauche à droite.
- On peut aussi écrire (1 désignant la sortie standard **stdout**) :
  - **`$ programme 1> fichier`**



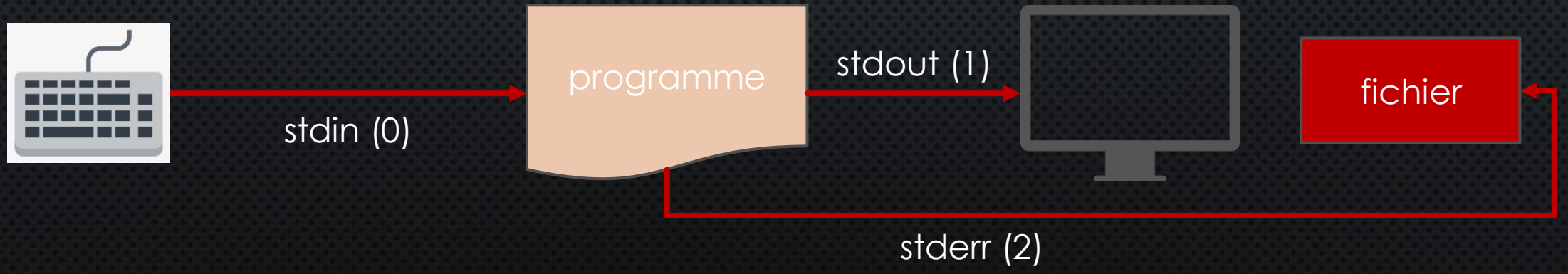
REDIRECTION DE LA SORTIE STANDARD

- Variante :
  - `$ programme >> fichier`
  - `$ programme 1>> fichier`
- L'opérateur >> ajoute la sortie standard au fichier (append) au lieu de l'écraser s'il existe déjà (sinon il le créé de la même façon).



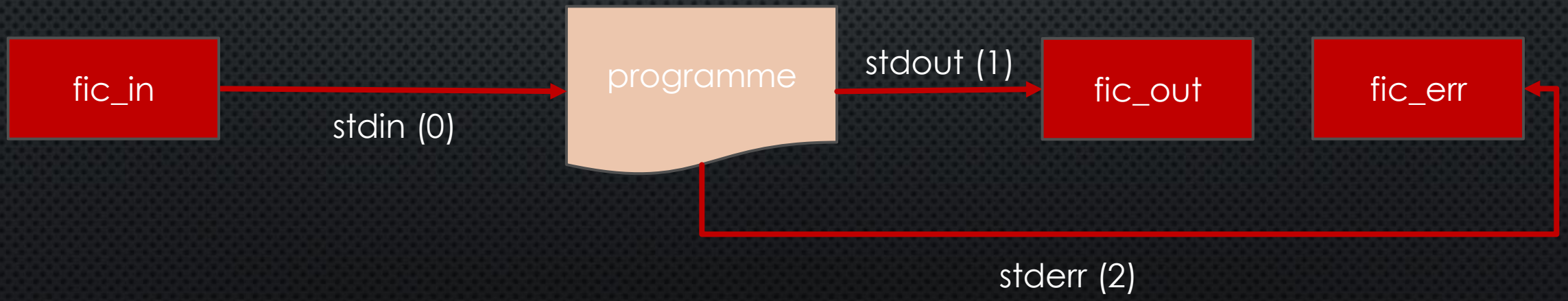
REDIRECTION DE LA SORTIE ERREUR STANDARD

- Lorsqu'un programme génère une erreur, il émet éventuellement un message sur une sortie spéciale : stderr (en plus de générer un code de retour non nul en fin de processus)
- On peut diriger la sortie **erreur** d'un programme vers un fichier ou un périphérique :
  - `$ programme 2> fichier`
- Ici aussi les données vont de gauche à droite.
- L'option **2>>** est également possible ici (mode append)



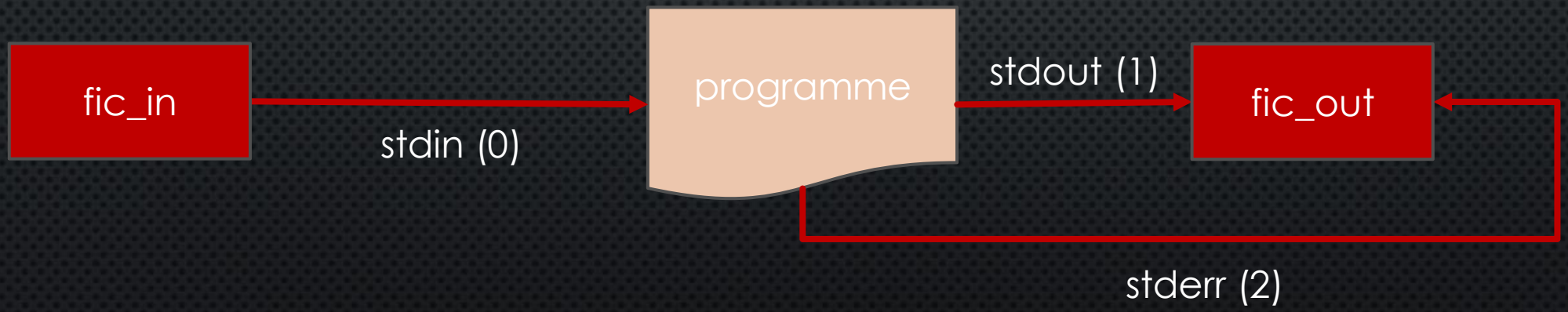
REDIRECTIONS STDIN, STDOUT, STDERR

- Les redirections peuvent bien entendu être combinées :
  - **`$ programme < fic_in >fic_out 2> fic_err`**



REDIRECTIONS STDIN, STDOUT, STDERR

- Ou encore :
  - **`$ programme < fic_in &>fic_out`**
- L'opérateur &> est équivalent à 2>&1



REDIRECTIONS - EXEMPLES

- La commande **cat** permet de concaténer des fichiers. Selon l'usage, elle va créer, copier, afficher, fusionner des fichiers.
- Essayer les commandes suivantes (depuis votre répertoire personnel) :
  - `$ cat > fic.txt`
  - `ceci`
  - `est un`
  - `fichier texte.`
  - `[CTRL] + [D]`
  - `$ cat fic.txt`
- Le programme cat « envoie » l'entrée standard vers le fichier fic.txt
- CTRL-D indique la fin du flux d'entrée.

REDIRECTIONS - EXEMPLES

- Essayer cette variante :
  - `$ cat << FIN >> fic.txt`
  - `ceci`
  - `est la`
  - `suite`
  - `FIN`
  - `$ cat fic.txt`
- L'opérateur << permet de créer une étiquette (label) : un mot-clé indiquant la fin de l'entrée standard.

REDIRECTIONS - EXEMPLES

- Entrez et analysez les commandes suivantes :

```
1. $ cat > fdisk.txt
2. p
3. F
4. [CRTL-D]
5. $ cat fdisk.txt
6. su
7. # fdisk /dev/?da < fdisk.txt
8. exit
```

REDIRECTIONS - EXEMPLES

- Entrez et analysez les commandes suivantes :
  1. `$ date > date.log`
  2. `$ cat date.log`
  3. `$ date >> date.log`
  4. `$ cat date.log`
  5. `$ date > date.log`
  6. `$ cat date.log`
- On voit ici la nuance entre > et >>

REDIRECTIONS - EXEMPLES

- Entrez et analysez les commandes suivantes :

```
1. $ ls /etc > ls.log
2. $ ls /fake > ls.log
3. $ cat ls.log
4. $ ls /fake 2> ls.log
5. $ ls /etc /fake > ls.log 2> ls.err
6. $ cat ls.log
7. $ cat ls.err
8. $ ls /etc /fake &> ls.log
9. $ cat ls.log
10.$ ls /etc > /dev/null
```

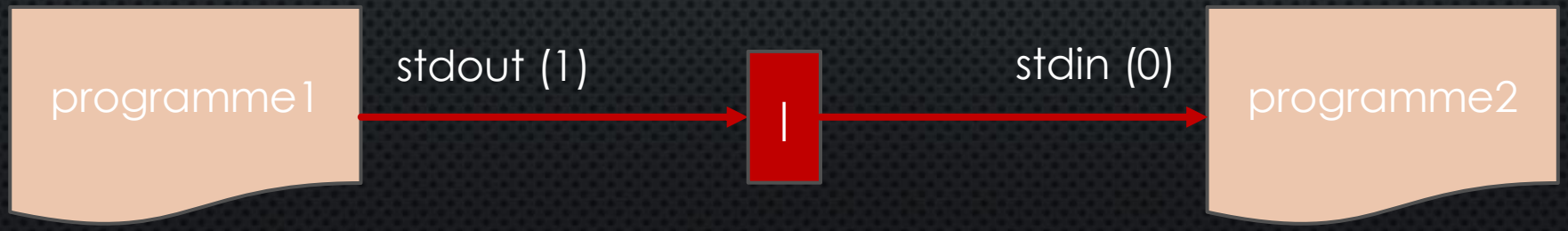
REDIRECTIONS - EXEMPLES

- Créez, analysez et exécutez le script suivant :

```
#!/usr/bin/env bash
for n in 1 2 3 a b c
do
touch fichier-$n.txt
cat << EOF > fichier-$n.txt
Ceci est le fichier n°$n
Ligne 2
Ligne 3
Ligne 4
EOF
echo "fichier-$n.txt créé"
done
```

LES TUBES (PIPES)

- Les tubes sont représentés par l'opérateur "|".
- Les données vont de gauche à droite.
- La figure suivante indique comment la sortie standard du premier processus est redirigée vers l'entrée standard du second processus.
  - **`$ programme1 | programme2`**



LES TUBES (PIPES)

- Exemples :
  - `$ apt list`
  - `$ apt list | grep python`
  - `$ apt list | grep ^python`
  - `$ ps aux`
  - `$ ps aux | grep login`

CAT : L'ÉDITEUR RUDIMENTAIRE

- La commande **cat** peut être utilisée comme un éditeur de texte rudimentaire.
  - **`$ cat > texte.txt`**
  - **`ligne 1`**
  - **`ligne 2`**
  - **`ligne 3`**
  - **`[CTRL]+[D]`**
- Vous noterez l'utilisation de **[ctrl] + [D]** : cette combinaison est utilisée pour clore la saisie, et plus généralement pour sortir du processus courant.

CAT : LECTEUR DE TEXTE

- On utilise plus couramment cat pour envoyer du texte vers la sortie standard.
- Les options les plus courantes sont :
  - -n : numéroter chaque ligne de la sortie
  - -b : numéroter uniquement les lignes non vides
  - -A : afficher le retour chariot ($) et autres caractères non-imprimables
- Exemple :
  - **`$ cat -n /etc/resolv.conf`**

TAC : LECTEUR INVERSE

- **tac** fait la même chose que **cat** à l'exception qu'elle lit de la dernière ligne à la première.
- Par ailleurs, les options ne sont pas les mêmes.
  - **`$ tac texte.txt`**

```
ligne 3
ligne 2
ligne 1
```

COMMANDES HEAD ET TAIL

- On utilise souvent les commandes **head** et **tail** pour analyser les fichiers de journaux.
- Par défaut, ces commandes affichent 10 lignes.
- En voici les utilisations les plus courantes :
  - afficher les 20 premières lignes de /var/log/messages :
    - **`$ head -n 20 /var/log/messages`**
    - **`$ head -20 /var/log/messages`**
  - afficher les 20 dernières lignes de /var/log/messages :
    - **`$tail -20 /var/log/messages`**

COMMANDE TEE

- La commande **tee** permet à la fois de lire un flux et de le rediriger.
- Par exemple, **tee** donne la sortie **et** l'écrit dans le fichier ls1.txt :
  - `$ ls | tee ls1.txt`
- Analyser et essayer cet exemple simple :
  - `$ tee test << EOF`
  - `$ cat test`

MANIPULATION DE TEXTE

- Voyons quelques outils de base permettant de manipuler du texte, notamment :
  - Compter des lignes, des mots, des octets
  - Afficher les fichiers binaires
  - Découper les fichiers
  - Trouver des doublons
  - Trier la sortie
  - Couper des fichiers
  - Jointure de texte
  - Convertir les caractères
  - …

LA COMMANDE WC

- La commande **wc** compte le nombre d'octets, de caractères, de mots et de lignes dans les fichiers.
- Les options suivantes permettent de sélectionner ce qui nous intéresse :
  - **-l** compte le nombre de lignes
  - **-w** compte le nombre de mots (words)
  - **-c** compte le nombre d'octets
  - **-m** compte le nombre de caractères
  - **-L** indique le nombre de caractères de la ligne la plus longue
- sans argument, wc compte ce qui est saisi dans **stdin**.

LA COMMANDE WC

- Quelques exemples à essayer :
  - **`$ wc -l /etc/passwd`**
  - **`$ cat /etc/passwd | wc -l`**
- Exercice : compléter la commande ci-dessous pour que la variable C contienne le nombre d'entrées du répertoire **/etc/** :
  - **`$ X=$(ls       |       )`**

LA COMMANDE SPLIT

- La commande **split** découpe un fichier en plusieurs fichiers plus petits à partir de critères comme la taille ou le nombre de lignes.
- Par exemple, nous pouvons découper **/etc/passwd** en fichiers de 5 lignes chacun :
  - **`$ split -l 5 /etc/passwd`**
- Cette commande va créer des fichiers appelés xaa, xab, xac, xad, etc., chaque fichier contenant au plus 5 lignes. Essayez cette variante :
  - **`$ split -dl 5 /etc/passwd test`**
- En déduire le rôle de l'option -d et du second argument.

LA COMMANDE CUT

- La commande **cut** permet d'extraire une plage de caractères ou de champs de chaque ligne d'un fichier texte.
- Par exemple :
  - `$ cut -c5-10,15- /etc/passwd`
- Cette commande extrait les caractères 5 à 10 puis 15 jusqu'à la fin pour chaque ligne de **/etc/passwd**.

LA COMMANDE CUT

- On peut spécifier le séparateur de champ (espace, virgule, etc.) d'un fichier ainsi que les champs à extraire. Ces options sont définies respectivement par les options **-d** (delimiter) et **-f** (field).
  - `$ cut -d: -f 1,7 /etc/passwd`
- Cette commande extrait les 1er et 7ème champs de **/etc/passwd** séparés par un espace. Le délimiteur de sortie est le même que le délimiteur d'entrée d'origine (par défaut, la tabulation).
- L'option **--output-delimiter** permet de changer le délimiteur de sortie :
  - `$ cut -d: -f 1,7 --output-delimiter=" " /etc/passwd`

LA COMMANDE UNIQ

- La commande **uniq** permet d'éliminer les lignes successives en doublon.
- Elle n'envoie sur **STDOUT** qu'une version de lignes successives identiques.
- Voyons avec un exemple.
- Créons un fichier contenant différentes lignes avec des doublons :

LA COMMANDE UNIQ

- **$ cat > /tmp/maliste**
- **Ligne 1**
- **Ligne 1**
- **Ligne 2**
- **Ligne 3**
- **Ligne 3**
- **Ligne 3**
- **…**
- **Ligne 1**
- **Ligne 2**
- **Ligne 2**
- **[CTRL] + [D]**

Ou encore :

```
$ touch /tmp/maliste
$ for i in {1..20}
> do
> echo "Ligne $(($RANDOM%3+1))" >> /tmp/maliste
> done
```

LA COMMANDE UNIQ

- Visualisons notre fichier :
  - **`$ cat /tmp/maliste`**
- Avec la commande **uniq** :
  - **`$ uniq /tmp/maliste`**
- Ou bien sous cette forme :
  - **`$ cat /tmp/maliste | uniq`**

LA COMMANDE SORT

- La commande **sort** permet de trier les lignes d'un fichier.
- Par défaut, **sort** trie le texte par ordre alphabétique. Pour effectuer un tri numérique, il faut utiliser l'option **-n**.
- Appliquons la commande sur notre précédent exemple :
  - `$ sort /tmp/maliste`
- Ou bien sous cette forme :
  - `$ cat /tmp/maliste | sort`

LA COMMANDE SORT

- Et admirons les effets sympathiques des combinaisons :
  - `$ sort /tmp/maliste | uniq`
- Ou bien encore :
  - `$ cat /tmp/maliste | sort | uniq`
- Et même, pour finir :
  - `$ cat /tmp/maliste | sort | uniq > /tmp/maliste_nette`

LA COMMANDE TR

- La commande **tr** convertit « à la volée » un ensemble de caractères en un autre.
- Cette commande ne prend que deux arguments : caractères initiaux, caractères de remplacement. Le nom du fichier n'est pas un argument, il faut donc utiliser **tr** comme un filtre.
- Exemple :
  - **`$ tr 'a-z' 'A-Z' < /tmp/maliste`**
- Ou bien cette mini-crypto parfaitement symétrique :
  - **`$ tr 'a-m,n-z' 'n-z,a-m' > /tmp/msg_secret`**

AUTRES COMMANDES

- Voici quelques autres commandes de manipulation de texte :
- Remplacer les tabulations par des espaces
  - On utilise la commande **expand** pour remplacer les tabulations par des espaces.
  - **unexpand** est utilisé pour l'opération inverse.
- Afficher les fichiers binaires
  - Il y a nombre d'outils pour cela ; le plus courant est **hexdump**
  - **`$ hexdump /etc/hostname`**
- Voir aussi les commandes : **paste**, **join**, **fmt**, **pr**.

#4

OUTILS AVANCÉS

DE TRAITEMENT DU TEXTE

LA COMMANDE GREP

- **grep** recherche dans les fichiers d'entrée indiqués les lignes correspondant à un certain motif.
  - `$ grep [OPTIONS] MOTIF [FICHIER]`
- Si aucun fichier n'est fourni, ou si le nom "-" est mentionné, la lecture se fait depuis l'entrée standard (STDIN)
- Par défaut, **grep** affiche les lignes correspondant au MOTIF.
- Consulter l'aide de **grep** pour en savoir plus sur les (nombreuses) options disponibles ($ man grep)

LA COMMANDE GREP

- Il existe trois variantes principales de **grep**, contrôlées par les options suivantes.
  - **-G** : Interprète le motif comme une expression régulière simple. C'est le comportement par défaut.
  - **-E** : Interprète le motif comme une expression régulière étendue.
  - **-F** : Interprète le motif comme une liste de chaînes figées, séparées par des Sauts de Lignes (NewLine). La correspondance est faite avec n'importe laquelle de ces chaînes.
- De plus, il existe deux variantes du programme : **egrep** et **fgrep**.
  - **egrep** est similaire (sans être identique) à **grep -E**
  - **fgrep** est identique à **grep -F**.

LES EXPRESSIONS RÉGULIÈRES

- En informatique, une **expression régulière** (ou **expression rationnelle**, plus francophone) ou encore **motif**,
- est une chaîne de caractères, qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles.
- Les expressions régulières sont également appelées **regex** (de l'anglais *regular expression*).
- Exemple :
  - `^[[:digit:]]+ -[[:blank:]]+.*$`

LES EXPRESSIONS RÉGULIÈRES

- Afin d'expérimenter les expressions régulières, nous allons utiliser un script.
- Le script **regexp.sh** ci-contre compare une expression rationnelle à des chaînes de caractères et donne le résultat.
- Il est recommandé pour bien le comprendre de le saisir manuellement,
- Il est cependant aussi récupérable en ligne de commande.
- Il ne faut pas oublier de le rendre exécutable.
- Et enfin, créons un alias « re » pour simplifier l'appel au script.

```
1  #! /bin/sh
2  # source : Christophe Blaess, Scripts Shell Linux
3  EXPRESSION="$1"
4  # Eliminons l'expression des arguments de ligne de commande :
5  shift
6  # Puis comparons-la avec les chaines :
7  for chaine in "$@"
8  do
9  echo "$chaine" | grep -E "$EXPRESSION" > /dev/null
10 if [ $? -eq 0 ]
11 then
12 echo "$chaine : OUI"
13 else
14 echo "$chaine : NON"
15 fi
16 done
```

```
$ wget http://melot.fr/div/regexp.sh
$ chmod u+x regexp.sh
$ alias re="./regexp.sh"
```

LES EXPRESSIONS RÉGULIÈRES

- Essayons quelques exemples :
- Lesquels des trois mots « Bonjour », « ou » et « bonsoir » vérifient l'expression régulière « ou » :
  - **`$ re ou Bonjour ou bonsoir`**
- Quelques nuances :
  - **`$ re ou Bonjour oU bonsoir`**
  - **`$ re o. Bonjour oU bonsoir`**

LES EXPRESSIONS RÉGULIÈRES

- Découvrons quelques opérateurs regex :
- Le symbole générique .
  - **`$ re o.r Bonjour oU bonsoir`**
  - **`$ re o.r Bonjour oU bonsoiiiiiir`**
  - **`$ re o.r Bonjour oU bonsoor`**

LES EXPRESSIONS RÉGULIÈRES

- Début (^) et fin ($) de chaîne :
  - **`$ re '^B' Bonjour oU bonsoor`**
  - **`$ re 'r$' Bonjour oU bonsoor`**
  - **`$ re '^oU$' Bonjour oU bonsoor`**
  - **`$ re '^ou$' Bonjour oU bonsoor`**
  - **`$ re 'ou.$' Bonjour oU bonsoor`**

LES EXPRESSIONS RÉGULIÈRES

- Alternatives | :
  - **`$ re 'ou|oi' Bonjour ou bonsoir`**
  - **`$ re 'ou|oi' Bonjour ou bonsoor`**
  - **`$ re 'ou|oi|oo' Bonjour ou bonsoor`**
- *NB : dans le cas d'une expression régulière simple, l'opérateur alternatif* ***« | »*** *doit être précédé du caractère d'échappement « \ » . Il en va de même pour un certains nombre d'opérateurs, ce qui alourdit quelque peu l'écriture.*

LES EXPRESSIONS RÉGULIÈRES

- Listes [ ] :
  - **$ re '[ji]' Bonjour ou bonsoir**
  - **$ re 'n[ji]' Bonjour ou bonsoir**
  - **$ re 'n[js]' Bonjour ou bonsoir**

LES EXPRESSIONS RÉGULIÈRES

- Intervalles [-] :
  - **`$ re 'o[a-z]' Bonjour o5 bonsoir`**
  - **`$ re '[A-Z]o' Bonjour o5 bonsoir`**
  - **`$ re 'o[0-9]' Bonjour o5 bonsoir`**
- Le symbole ^ au début d'un intervalles ou d'une liste indique la négation :
  - **`$ re 'n[js]' Bonjour ou bonsoir`**
  - **`$ re '[^A-Z]on' Bonjour bonsoir Bon`**
  - **`$ re 'o[^0-9]' Bonjour o5 bonsoir`**

LES EXPRESSIONS RÉGULIÈRES

- Les classes sont plus commodes à utiliser que les intervalles
- Voici les douze classes standards :

Nom	Signification	Exemple en Ascii
alpha	Lettres alphabétiques dans la localisation en cours.	[A-Za-z]
digit	Chiffres décimaux.	[0-9]
xdigit	Chiffres hexadécimaux.	[0-9A-Fa-f]
alnum	Chiffres ou lettres alphabétiques.	[[:alpha:][:digit:]]
lower	Lettres minuscules dans la localisation en cours.	[a-z]
upper	Lettres majuscules dans la localisation en cours.	[A-Z]
blank	Caractères blancs.	espace et tabulation
space	Caractères d'espacement.	espace, tabulation, sauts de ligne et de page, retour chariot
punct	Signes de ponctuation.	[]!"#$%&'()*+,-./:;<=>?@\ ^_`{}~[]
graph	Symboles ayant une représentation graphique.	[[:alnum:][:punct:]]
print	Caractères imprimables (graph et l'espace).	[[:graph:]]
cntrl	Caractères de contrôle.	Codes Ascii inférieurs à 31, et caractère de code 127

LES EXPRESSIONS RÉGULIÈRES

- Les classes se notent dans le format **`[[:classe:]]`**
- Exemple :
  - **`$ re "[[:punct:]]" bonjour bonjour,`**

LES EXPRESSIONS RÉGULIÈRES

- Opérateurs de répétition :

*****	toute occurrence de l'élément précédent même l'absence
+	une ou plusieurs occurrence de l'élément précédent
?	zéro ou une occurrence de l'élément précédent
{n,m}	au moins n et au plus m occurrences de l'élément précédent

LES EXPRESSIONS RÉGULIÈRES

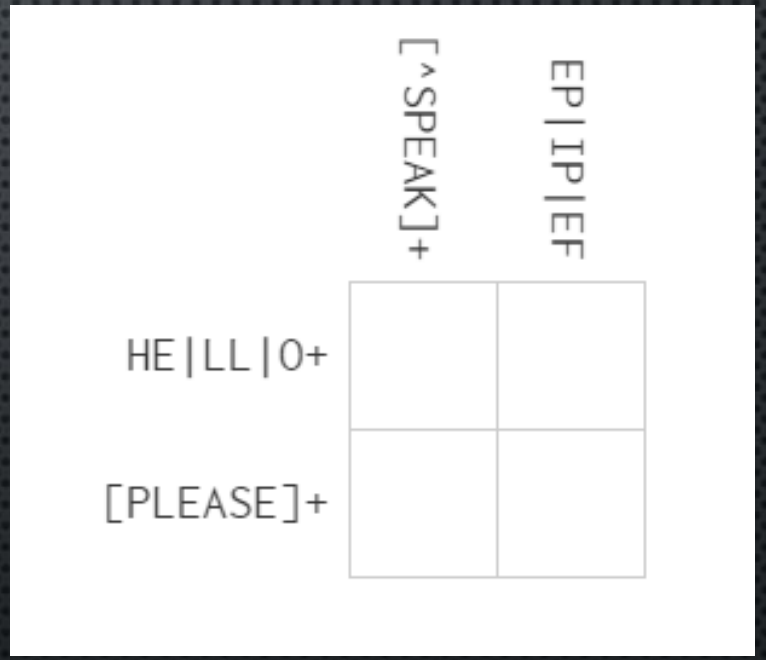
- Exemples de répétitions :
  - Tout caractère :
    - **`$ re "b.*" b bo bon bonjour Bonjour`**
    - **`$ re "bo*n" bn bon booooooon`**
  - Une ou plusieurs occurrences :
    - **`$ re "bo+n" bn bon booooooon`**
  - Aucune ou une occurrence :
    - **`$ re "bo?n" bn bon booooooon`**
  - Un minimum d'occurrences :
    - **`$ re "bo{2,}n" bn bon booooooon`**
  - Un maximum d'occurrences :
    - **`$ re "bo{0,2}n" bn bon booooooon`**
  - Un nombre exact d'occurrences :
    - **`$ re "bo{2}n" bn bon booooooon`**

LES EXPRESSIONS RÉGULIÈRES

- **Groupements () :**
- Un groupement permet de rechercher une chaine précise, et sa répétition, ici au minimum deux fois à la suite :
  - `$ re "(bon){2}" bon bonbon`
- La référence « arrière » \n :
  - `$ re "(bon)\1" bon bonbon bonbonbon`
- Elle permet de répliquer le nième groupe du motif (ici le premier, soit : (bon)

LES EXPRESSIONS REGULIERES

- Résoudre cette grille sur le thème « Beatles » (une lettre par case)

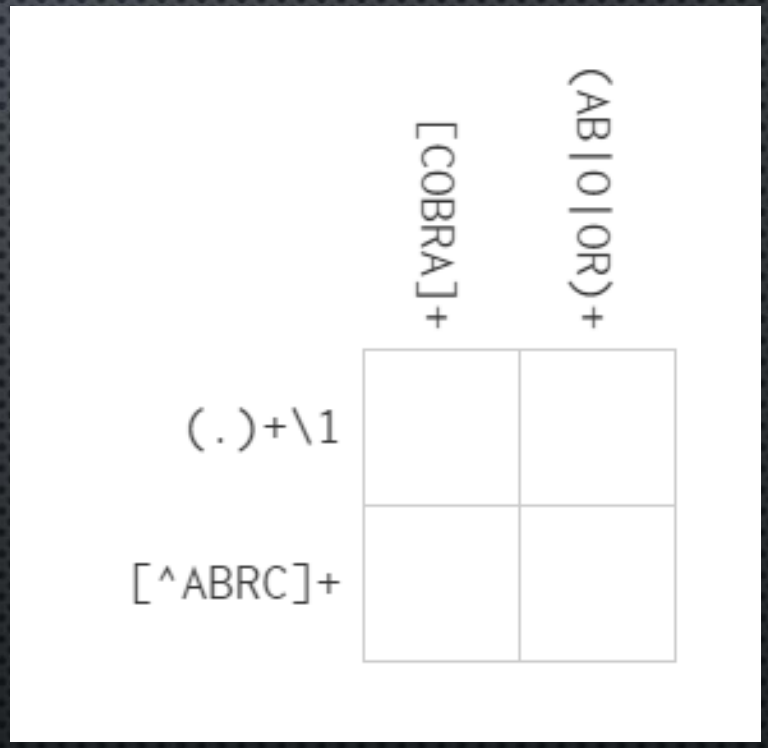


LES EXPRESSIONS REGULIERES

- Résoudre cette grille (une lettre par case)

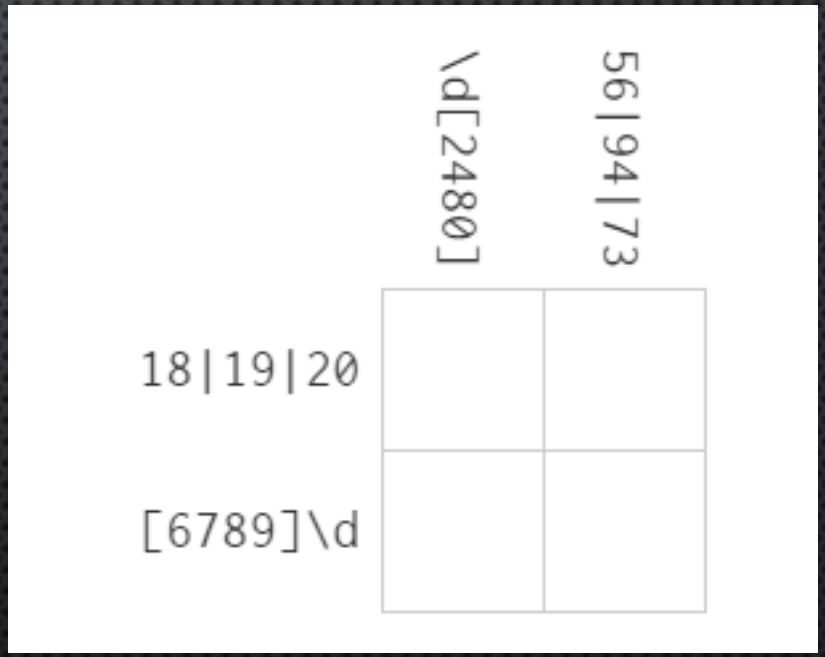
LES EXPRESSIONS REGULIERES

- Résoudre cette grille (une lettre par case)



LES EXPRESSIONS REGULIERES

- Résoudre cette grille (une lettre par case)
  - NB : **\d** est un raccourci pour **[0-9]**
  - *Notation non conforme POSIX*
  - *Il faut utiliser **grep -P** pour accepter cette notation (P pour Perl)*
  - *Il existe ainsi \w, \s, …*



Source : https://regexcrossword.com/

AUTRES OUTILS

- Il existe d'autres outils qui permettent de traiter du texte de façon avancée, grâce notamment aux expression régulières.
- Les outils les plus habituels sont **AWK** et **SED**
- Ils seront abordés en détail plus loin dans ce cours, ainsi que l'utilitaire **find** qui permet d'effectuer des recherches poussées dans le système de fichiers.
- Cependant, ils pourront apparaître dans certains exercices, en mode « découverte ».

EXERCICES GREP

- Extraire l'adresse IPv4 de l'interface ens32 (le nom de l'interface est à modifier selon votre environnement).
- Essayez et analysez cette progression :
  - `$ ip -4 addr show ens32`
  - `$ ip -4 addr show ens32 | grep inet`
  - `$ ip -4 addr show ens32 | grep inet | awk '{ print $2; }'`
  - `$ ip -4 addr show ens32 | grep inet | awk '{ print $2; }' | cut -d/ -f1`

EXERCICES GREP

- Pour retirer les lignes de commentaires :
  - **`$ grep -v "^#" /etc/network/interfaces`**
- Pour retirer en plus les lignes vides :
  - **`$ grep -v "^#" /etc/network/interfaces | grep -v "^$"`**
- Ou encore :
  - **`$ grep -vE "^#|^$" /etc/network/interfaces`**
- Ou mieux encore :
  - **`$ grep -vE "^\s*#|^$" /etc/network/interfaces`**
- Analyser et expliquer cette dernière commande.

Pour en savoir plus sur grep et les regex :
https://www.gnu.org/software/grep/manual/html_node/index.html

EXERCICES GREP

- Repérez le fichier **/usr/share/dict/words**
- Que contient ce fichier ?
- Combien y a-t-il de mots ?
- Combien y a-t-il de mots contenant « magi » ?
- Combien y a-t-il de mots commençant par « magi » ?
- Combien y a-t-il de mots se terminant par « pire » ?
- Quels sont les mots qui commencent par un « z » et qui contiennent un « ï » ?
- Quels sont les mots qui ne se terminent pas par une lettre ?
- Ecrire un script pour créer 26 fichiers contenant tous les mots commençant par chaque lettre de l'alphabet, de a à z (les fichiers seront appelés wa.txt, wb.txt, …, wz.txt)

L'ÉDITEUR DE TEXTE VI

- **vi** est un éditeur de texte en mode texte plein écran écrit par Bill Joy en 1976 sur une des premières versions de la distribution Unix BSD.
- vi - ou l'un de ses clones - peut être trouvé dans presque toutes les installations de Unix.
- Les utilisateurs débutants avec **vi** sont souvent confrontés à des difficultés, d'une part à cause des raccourcis utilisés pour chacune des commandes, ensuite parce que l'effet de ces raccourcis change selon le mode dans lequel se trouve vi.
- On installe vi Improved (vim) :
  - `# yum install vim || apt install vim`
- On pourra changer l'éditeur par défaut sous Debian/Ubuntu avec la commande :
  - `$ sudo update-alternatives --config editor`

L'ÉDITEUR DE TEXTE VI

- Pour utiliser parfaitement vi, il faut simplement s'exercer en utilisant le « guide de survie VI » associé à ce cours (ou encore l'article wikipedia sur le sujet)
- A force de pratique, la manipulation devient naturelle.
- Une alternative plus simple consiste à utiliser l'éditeur **nano :**
  - **`# yum install nano || apt-get install nano`**
- A chacun ici de faire son choix…

#5

Le système de fichiers

Filesystem Hierarchy Standard (FHS)

LA STRUCTURE DU SYSTÈME DE FICHIERS

- Un système de fichiers (notamment sous Unix) est similaire à une arborescence,
- avec une racine qui se scinde en branches et sous-branches, soit en répertoires et sous-répertoires.
- On commence par le tronc principal, la racine (root) : /
- L'analogie avec **C:** de Windows NT a ses limites : C:\ désigne le premier périphérique de stockage de masse, alors que la racine Unix peut correspondre à n'importe quel disque (partition) de votre système (point de montage).
- La racine contient différents répertoires et sous-répertoires contenant eux-mêmes des fichiers.
- Le système des fichiers est « unifié » : tout part de la même racine.

LA COMMANDE TREE

- La commande **tree** liste le contenu de répertoires sous forme d'arborescence.
  - **`# yum install -y tree || apt install tree`**
- Par exemple sous Centos 7, l'arborescence à partir de la racine :
  - **`$ tree -L 1 /`**
- Quelles différences avec Debian ?

```
$ tree -L 1 /
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── srv
├── sys
├── tmp
├── usr
└── var

20 directories, 0 files
```

PARTITION RACINE

- Les répertoires suivants peuvent être montés sur d'autres partitions que la celle de la racine :
  - /boot
  - /home
  - /root
  - /tmp
  - /usr
  - /usr/local
  - /opt
  - /var

PARTITION RACINE

- Les répertoires :
  - **/dev, /bin, /sbin, /etc et /lib**
- **doivent être montés sur la partition racine.**
- De plus, la racine doit contenir un répertoire **/proc**
- Il est utilisé par le noyau pour informer sur le statut du système d'exploitation (processus, statistiques d'utilisation de la mémoire, etc.).

CONTENU DU SYSTÈME DE FICHIERS

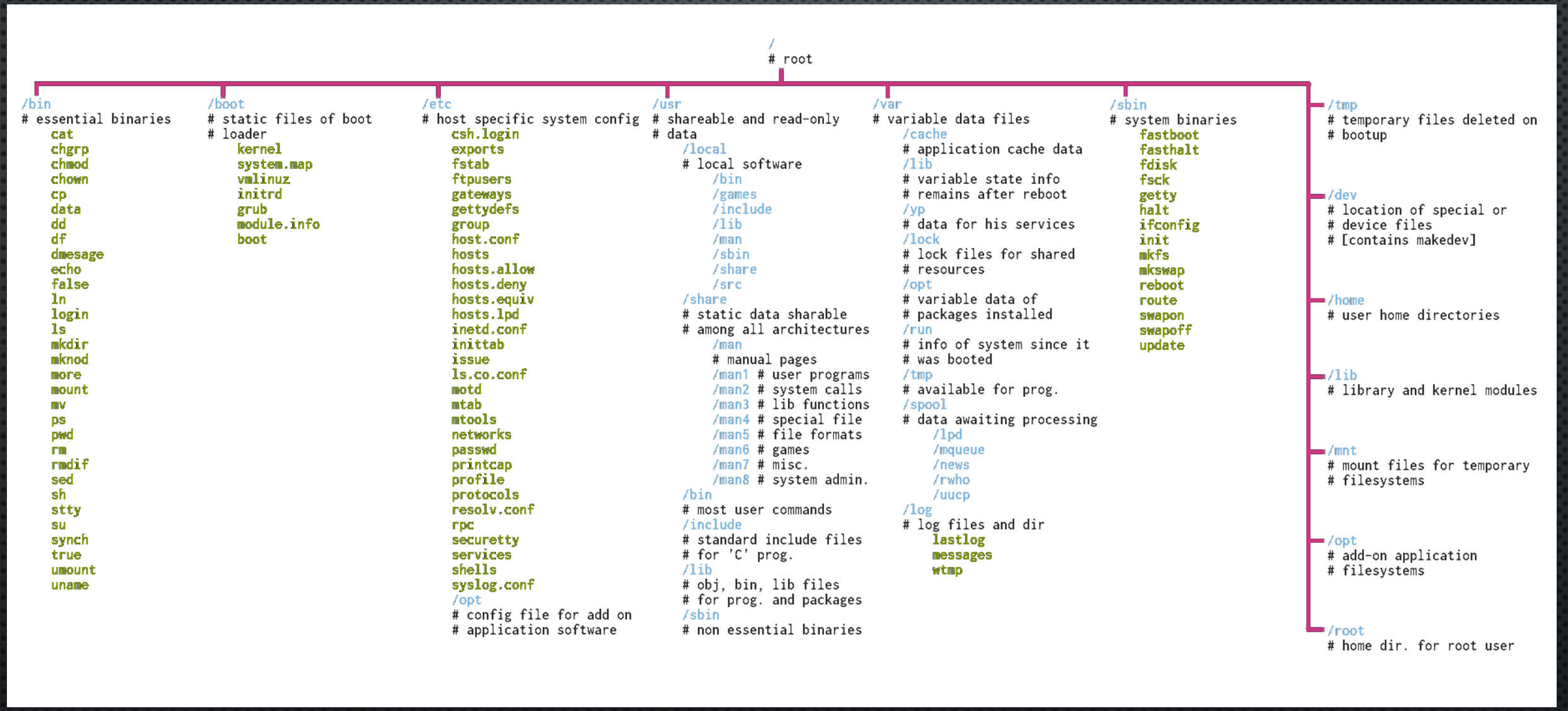
- /bin et /sbin : contiennent les **binaires** nécessaires au démarrage et les commandes essentielles (sbin pour les outils d'administration)
- /dev : « fichiers » **périphériques** ou fichiers spécifiques
- /etc : fichiers et répertoires de **configuration** spécifiques à la machine
- /lib et /lib64: **bibliothèques** partagées pour les binaires de /bin et /sbin. Contient également les modules du noyau.
- /mnt ou /media : points de montage pour les systèmes de fichiers **externes**
- /proc : **informations** du noyau. En lecture seule sauf pour /proc/sys.

CONTENU DU SYSTÈME DE FICHIERS

- /boot : contient le noyau Linux, les fichiers de **démarrage**, le System.map (carte des symboles du noyau) et les chargeurs d'amorçage secondaires.
- /home : contient les répertoires **utilisateurs**, avec, en général, une copie de /etc/skel.
- /root : répertoire de **l'utilisateur root**.
- /sys : export d'information du noyau, à la manière de /proc
- /tmp : fichiers temporaires.
- /usr : User Specific Ressource. Contenu essentiellement statique et partageable, composé de sous-répertoires bin, sbin, lib et autres qui contiennent des programmes et bibliothèques non essentielles ni nécessaires au démarrage.
- lost+found : est un dossier spécial de récupération des données du système de fichiers.

CONTENU DU SYSTÈME DE FICHIERS

- /usr/local ou /opt : programmes et bibliothèques supplémentaires. En général, c'est dans ces répertoires que l'on place les programmes qui ne font pas partie des paquets des distributions.
- /var : données variables comme les spool ou les journaux. Les sous-répertoires peuvent être soit partageables (comme /var/spool/mail) soit non partageables (comme /var/log).
- /var/www, /var/ftp ou /srv : pages web ou fichiers ftp anonymes.


/
root
/bin
essential binaries
cat
chgrp
chmod
chown
cp
data
dd
df
dmesage
echo
false
ln
login
ls
mkdir
mknod
more
mount
mv
ps
pwd
rm
rmdif
sed
sh
stty
su
synch
true
umount
uname
/boot
static files of boot
loader
kernel
system.map
vmlinuz
initrd
grub
module.info
boot
/etc
host specific system config
csh.login
exports
fstab
ftpusers
gateways
gettydefs
group
host.conf
hosts
hosts.allow
hosts.deny
hosts.equiv
hosts.lpd
inetd.conf
inittab
issue
ls.co.conf
motd
mtab
mtools
networks
passwd
printcap
profile
protocols
resolv.conf
rpc
securetty
services
shells
syslog.conf
/opt
config file for add on
application software
/usr
shareable and read-only
data
/local
local software
/bin
/games
/include
/lib
/man
/sbin
/share
/src
/share
static data sharable
among all architectures
/man
manual pages
/man1 # user programs
/man2 # system calls
/man3 # lib functions
/man4 # special file
/man5 # file formats
/man6 # games
/man7 # misc.
/man8 # system admin.
/bin
most user commands
/include
standard include files
for 'C' prog.
/lib
obj, bin, lib files
for prog. and packages
/sbin
non essential binaries
/var
variable data files
/cache
application cache data
/lib
variable state info
remains after reboot
/yp
data for his services
/lock
lock files for shared
resources
/opt
variable data of
packages installed
/run
info of system since it
was booted
/tmp
available for prog.
/spool
data awaiting processing
/lpd
/mqueue
/news
/rwho
/uucp
/log
log files and dir
lastlog
messages
wtmp
/sbin
system binaries
fastboot
fasthalt
fdisk
fsck
getty
halt
ifconfig
init
mkfs
mkswap
reboot
route
swapon
swapoff
update
/tmp
temporary files deleted on
bootup
/dev
location of special or
device files
[contains makedev]
/home
user home directories
/lib
library and kernel modules
/mnt
mount files for temporary
filesystems
/opt
add-on application
filesystems
/root
home dir. for root user

CHEMINS RELATIFS ET ABSOLUS

- On peut accéder à un répertoire ou un fichier en donnant son chemin complet, qui commence à la racine (/), ou en donnant son chemin relatif partant du répertoire courant.
- Chemin absolu :
  - indépendant du répertoire courant de l'utilisateur (ni de son répertoire personnel)
  - commence par /
- Chemin relatif :
  - dépend de l'endroit où se trouve l'utilisateur
  - ne commence pas par /

SE DÉPLACER DANS LE SYSTÈME DE FICHIERS

- Comme pour tout système de fichiers structuré, un certain nombre d'outils aident à parcourir le système.
- Les deux commandes suivantes sont des commandes internes du shell :
  - **pwd** : (Print Working Directory) affiche le répertoire actuel en chemin absolu
  - **cd** : la commande pour changer de répertoire (Change Directory)

EMPLACEMENTS

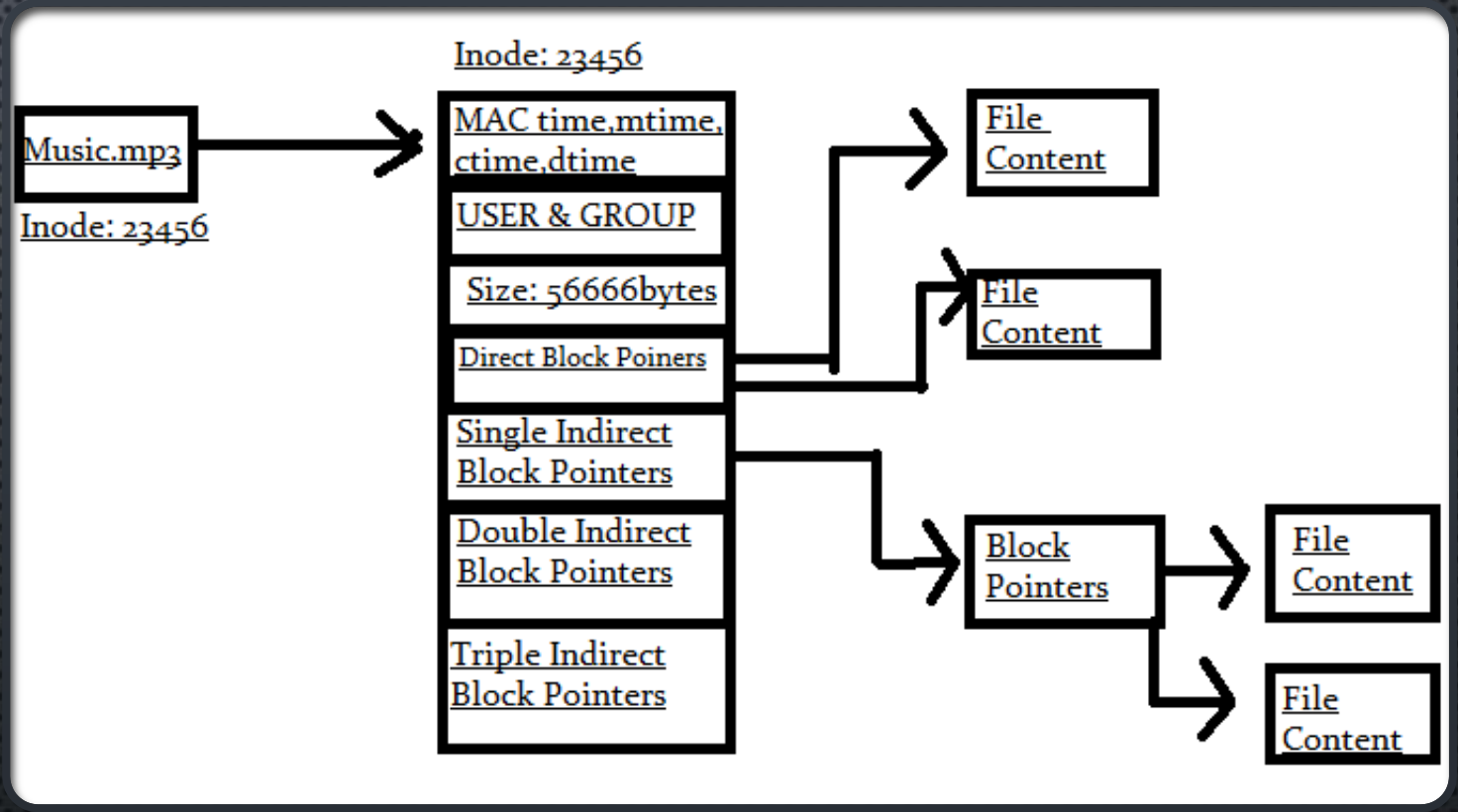
- L'emplacement courant est représenté par un point .
- L'emplacement parent est représenté par deux points ..
- Le répertoire utilisateur courant est représenté par le tilde ~

Le système de fichiers

Opérations sur les Fichiers

NOTION D'INODE

- Un inode est un **numéro unique** qui référence un ou plusieurs fichiers dans le système de fichiers.
- L'inode représente la **réalité physique** du fichier auquel un ou des noms (des emplacements sur le système de fichiers) correspondent.
- Les inodes contiennent notamment les métadonnées des fichiers, et en particulier celles concernant les **droits d'accès**.



LA COMMANDE LS

- La commande **ls** liste les fichiers (par défaut le répertoire courant)
- La commande **ls -l** affiche par défaut une liste triée par ordre alphabétique.
- Les commandes **dir** et **vdir** sont des variantes de ls.

```
$ ls -lah /etc/
total 1,4M
drwxr-xr-x. 122 root root   8,0K 11 jan 19:00 .
drwxr-xr-x.  18 root root   4,0K 11 jan 19:00 ..
-rw-r--r--.   1 root root     16  8 déc 20:44 adjtime
-rw-r--r--.   1 root root   1,5K  7 jun  2013 aliases
-rw-r--r--.   1 root root    12K  8 déc 20:48 aliases.db
drwxr-xr-x.   2 root root   4,0K  8 déc 21:04 alternatives
```

LA COMMANDE LS

```
$ ls -lah /etc/
total 1,4M
drwxr-xr-x 122 root root   8,0K 11 jan 19:00 .
drwxr-xr-x  18 root root   4,0K 11 jan 19:00 ..
-rw-r--r--   1 root root     16  8 déc 20:44 adjtime
-rw-r--r--   1 root root   1,5K  7 jun  2013 aliases
-rw-r--r--   1 root root    12K  8 déc 20:48 aliases.db
drwxr-xr-x   2 root root   4,0K  8 déc 21:04 alternatives
```

- Elle donne des attributs POSIX :
  1. type de fichier : - fichier, **d** répertoire, **l** lien symbolique
  2. droits : -, r, w et x, propriétaire (u), groupe (g) et autres (o).
  3. nombre de liens physique
  4. Propriétaire (u), ici root
  5. groupe propriétaire (g), ici root
  6. taille du fichier, ici en octets
  7. date de modification,
  8. nom du fichier.

LA COMMANDE LS

- Options de la commande ls :
  - -a : affiche aussi les fichiers cachés (dont le nom commence par .)
  - -r : tri inversé
  - -t : tri sur la date de modification (du plus récent au plus ancien)
  - -u : tri sur la date d'accès avec l'option -lt
  - -c : tri sur la date de changement de statuts des fichiers avec l'option -lt
  - -l : présentation tabulaire détaillée
  - -1 : présentation en liste continue
  - -i : affiche le numéro d'inode
- On trouvera encore plus d'options et détails dans la page man : man ls

CRÉATION D'UN RÉPERTOIRE

- Un répertoire (ou dossier) est une liste de descriptions de fichiers.
- Du point de vue du système de fichiers, il est traité comme un fichier dont le contenu est la liste des fichiers référencés.
- Un répertoire a donc les mêmes types de propriétés qu'un fichier comme le nom, la taille, la date, les droits d'accès et les divers autres attributs.
- La commande qui permet de créer des répertoires est **mkdir**

CRÉATION D'UN RÉPERTOIRE

- Une autre option couramment utilisée et utile de **mkdir** est **-p** qui crée les sous-répertoires quand c'est nécessaire.
- Par exemple (faites ces opération depuis votre ~) :

```
$ mkdir labs/ex1/script1
mkdir: impossible de créer le répertoire `labs/ex1/script1`: Aucun fichier ou répertoire de ce type
$ mkdir -p labs/ex1/script1
$ tree labs
labs
└── ex1
    └── script1
2 directories, 0 files
```

SUPPRESSION DES RÉPERTOIRES

- On utilise soit **rmdir** soit **rm -r** pour supprimer les répertoires.
- En tant que root, vous devrez peut-être spécifier l'option **-f** pour forcer la suppression de tous les fichiers.
- **rmdir** supprime un répertoire vide alors que **rm -r** supprime un répertoire et son contenu.
- Cet exemple illustre la suppression de tous les fichiers et sous-répertoires et laisse le répertoire **ex1** vide.

```
$ rm -rf labs/ex1/*
$ tree labs
labs
└── ex1

1 directory, 0 files
```

SUPPRESSION DES RÉPERTOIRES

- Cet autre exemple illustre la suppression de tous les fichiers et sous-répertoires y compris le répertoire ex1 lui-même :

```
$ rm -rf labs/ex1
$ tree labs
labs
```

COMMANDE TOUCH

- **touch** est une commande qui permet de créer et de modifier les fichiers.
- Syntaxe :
  - `$ touch {options} fichier(s)`
- Le fichier est créé s'il n'existait pas.
- Vous pouvez également changer la date d'accès avec l'option **-a**, la date de modification avec **–m**
- l'option **-r** copie les attributs de date d'un autre fichier.

COMMANDE TOUCH

- Exemples :
- Créer deux nouveaux fichiers **fic1.txt** et **fic2.txt** :

```
$ touch labs/fic1.txt labs/fic2.txt
$ ls -i labs
69308775 fic1.txt  69308776 fic2.txt
```

COMMANDE TOUCH

- Copier les attributs de date de **/etc/hosts** sur **fic2.txt** :

```
$ ls -l /etc/hosts
-rw-r--r--. 1 root root 83 10 déc 12:11 /etc/hosts
$ touch labs/fic2.txt -r /etc/hosts
$ ls -l labs/fic2.txt
-rw-rw-r--. 1 pascal pascal 0 10 déc 12:11 labs/fic2.txt
```

COMMANDE CP

- La commande **cp** permet de copier un ou plusieurs fichiers.
- Syntaxe :
  - `$ cp [options] fichier1 fichier2`
  - `$ cp [options] fichier(s) répertoire`
- Il est important de noter que **cp fichier1 fichier2** crée une nouvelle copie de fichier1 et laisse fichier1 inchangé.

COMMANDE CP

- Illustration :
- **fic1.txt** avec l'inode 69308775 est copié sur **fic3.txt**, en dupliquant les données sur un nouveau bloc et en créant une nouvelle inode 69308777 pour fic3.txt.

```
$ cp labs/fic1.txt labs/fic3.txt
$ ls -i labs
69308775 fic1.txt  69308776 fic2.txt  69308777 fic3.txt
```

COPIE RÉCURSIVE

- On peut également copier plusieurs fichiers dans un répertoire
- Exemple :
  - On ajoute un répertoire **rep1** dans **labs**
  - On copie tout le contenu de **labs** dans un dossier **bak** :
- Notons que le dossier de destination **bak** n'existait pas et a été créé.

```
$ mkdir labs/rep1
$ cp -r labs bak
$ tree bak labs
bak
├── fic1.txt
├── fic2.txt
├── fic3.txt
└── rep1
labs
├── fic1.txt
├── fic2.txt
├── fic3.txt
└── rep1
2 directories, 6 files
```

COPIE RÉCURSIVE

- Si on recommence l'opération alors que la destination existe déjà c'est le dossier source lui-même qui est copié :

```
$ cp -r labs bak
$ ls bak
fichier1.txt  fichier2.txt
fichier3.txt  labs  rep1
$ tree bak
bak
├── fichier1.txt
├── fichier2.txt
├── fichier3.txt
├── labs
│   ├── fichier1.txt
│   ├── fichier2.txt
│   ├── fichier3.txt
│   └── rep1
└── rep1

3 directories, 6 files
```

LA COMMANDE MV

- Syntaxe :
  - **`mv [options] anciennom nouveaunom`**
  - **`mv [options] source destination`**
  - **`mv [options] source répertoire`**
- La commande **mv** peut à la fois déplacer et renommer les fichiers et les répertoires :
  - Si **anciennom** est un fichier et **nouveaunom** un répertoire, le fichier **anciennom** est déplacé dans ce répertoire.
  - Si la **source** et la **destination** sont sur le même système de fichiers, alors le fichier n'est pas copié, mais les informations de l'inode sont mises à jour pour tenir compte du nouveau chemin.
- Les options les plus courantes de **mv** sont **-f** pour forcer l'écrasement et **-i** pour demander confirmation à l'utilisateur.

RENOMMER ET DÉPLACER

- Vérifiez les valeurs **inode** des fichiers avec **ls -i** , après chacune de ces opérations :
- Renommer le répertoire bak en bak2
  - **`$ mv bak bak2`**
- Déplacer le bak2 dans le répertoire labs
  - **`$ mv bak2 labs`**
- renommer le répertoire bak2 en bak
  - **`$ mv labs/bak2 labs/bak`**

LIENS PHYSIQUES

- Un lien physique est un nom supplémentaire pour **un même inode**.
- Ainsi, le nombre de références au fichier s'accroît de un à chaque nouveau lien physique créé.
- Un lien est établi avec la commande **ln**
- Dans l'exemple page suivante, notez que le nombre de références est de 2 et que les deux fichiers ont la même taille (ainsi que la même valeur inode, ce que vous pouvez vérifier avec **ls -i**).
- Dans les faits, ces deux fichiers sont toujours parfaitement identiques (puisqu'ils sont un et un seul)
- Les liens physiques ne peuvent être créés que sur le même système de fichier, et une même partition.

LIENS PHYSIQUES

```
$ ln labs/fic1.txt labs/fic5.txt
$ ls -li labs
total 0
69308779 -rw-rw-r-- 2 pascal pascal 0 21 fév 07:04 fic1.txt
69308780 -rw-rw-r-- 1 pascal pascal 0 21 fév 07:04 fic2.txt
69308781 -rw-rw-r-- 1 pascal pascal 0 21 fév 07:04 fic3.txt
69308779 -rw-rw-r-- 2 pascal pascal 0 21 fév 07:04 fic5.txt
 3739518 drwxrwxr-x 2 pascal pascal 6 21 fév 07:04 rep1
```

LIENS SYMBOLIQUES

- Un lien symbolique vers un fichier ou un répertoire crée un **nouvel inode** qui **pointe** vers le **même bloc de données**.
- C'est aussi la commande ln qui est utilisée.
- Les liens symboliques peuvent pointer vers des fichiers ou répertoires présents sur un autre système de fichier.

LIENS SYMBOLIQUES

```
$ cd labs
$ ln -s fic1.txt fic4.txt
$ cd ..
$ ls -li labs
total 0
69308779 -rw-rw-r-- 2 pascal pascal 0 21 fév 07:04 fic1.txt
69308780 -rw-rw-r-- 1 pascal pascal 0 21 fév 07:04 fic2.txt
69308781 -rw-rw-r-- 1 pascal pascal 0 21 fév 07:04 fic3.txt
69308774 lrwxrwxrwx 1 pascal pascal 12 21 fév 07:21 fic4.txt -> fic1.txt
69308779 -rw-rw-r-- 2 pascal pascal 0 21 fév 07:04 fic5.txt
 3739518 drwxrwxr-x 2 pascal pascal 6 21 fév 07:04 rep1
$ echo $(date) > labs/fic1.txt
$ cat labs/fic4.txt
jeu. fév. 21 07:22:01 CEST 2019
```

COPIE PAR BLOCS

- **dd** est une commande Unix permettant de copier un fichier (avec ou sans conversion au passage) notamment sur des périphériques « blocs » tels que des disques durs, des clés USB ou des lecteurs CD-ROM ou inversement.
- Contrairement à **cp**, la commande **dd** copie des portions de données brutes d'un périphérique.
- Par conséquent, **dd** préserve le système de fichier sous-jacent.
- Alors que **cp** transfère les données (fichiers) d'un système de fichier à un autre.

SYNTAXE DE LA COMMANDE DD

- La syntaxe de **dd** est différente des autres commandes Unix traditionnelles.
- **dd** utilise des options de la forme **option=valeur** au lieu des habituelles **-o** valeur ou **--option=valeur**.
- Les principales options de dd sont les suivantes :
  - **if**=fichier_entree (Input File) : lit ce fichier en entrée. Cela peut être un fichier régulier comme un périphérique de type bloc. Par défaut, c'est l'entrée standard qui est utilisée (par exemple le clavier).
  - **of**=fichier_sortie (Output File) : écrit dans ce fichier en sortie.
  - **bs**=t_b (Block Size) : copie les données par blocs de t_b octets.
  - **count**=n_b : ne copie que n_b blocs.
  - **skip**=n_e : ignore les n_e premiers blocs du fichier d'entrée1 (Ne copie le fichier d'entrée qu'à partir du bloc de rang n_e+1.)
  - **seek**=n_s : ignore les n_s premiers blocs du fichier de sortie1 (Ne commence à écrire dans le fichier de sortie qu'à partir du bloc de rang n_s+1.)

SYNTAXE DE LA COMMANDE DD

- Exemples :
- Créer l'image d'un disque :
  - `$ dd if=/dev/sdb of=~/sdb.img`
- Copier un disque sur l'autre :
  - `$ dd if=/dev/sdb of=/dev/sdc conv=noerror,sync`
- Copier une partition :
  - `$ dd if=/dev/sdb1 of=~/sdb1.img`

RECHERCHE DE FICHIER AVEC FIND

- Syntaxe :
  - `$ find <REPERTOIRE> <CRITERE> [-exec <COMMANDE> {} \;]`
- Paramètres principaux :
  - REPERTOIRE indique l'emplacement de démarrage de la recherche
  - CRITERE peut être (parmi d'autres), le nom du fichier ou répertoire recherché.
- *Pour allez plus loin, consulter l'aide de la commande **find** (man find) ainsi que celle de la commande **xargs**, souvent associée.*

RECHERCHE DE FICHIER AVEC FIND

- Exemples :
  - **`find /usr/share/doc -name "x*"`**
  - **`find / -user 1000`**
  - **`find / -user 1000 2> /dev/null`**
- L'option **`-ls`** offre une sortie plus lisible :
  - **`find / -user 1000 -ls 2> /dev/null`**
- On remarque que les lignes correspondantes sont listées sur la sortie standard.

RECHERCHE DE FICHIER AVEC FIND

- On peut également lancer une commande sur cette sortie, comme supprimer le fichier ou changer le mode de permission, en utilisant l'option -exec.
- Par exemple, pour copier tous les fichiers appartenant à l'utilisateur 2015 (à condition d'en avoir les droits) :
  - `# find / -type f -user 2015 -exec cp -a {} ~/backup \;`
- Dans cet exemple :
  - `\;` à la fin de la ligne termine la commande **`-exec`**
  - et `{}` remplace chaque ligne trouvée par la commande **`find`**.

AUTRES OUTILS DE RECHERCHE

- **Recherche de fichiers avec which**
  - `$ which nom_de_commande`
- **which** retourne le chemin complet de la commande "nom_de_commande" en parcourant les répertoires définis dans la variable PATH de l'utilisateur uniquement.
- **Recherche de fichiers avec whereis**
  - `$ whereis nom_de_fichier`
- Cette commande affiche le chemin absolu des sources, binaires des pages manuel pour les fichiers correspondant à "nom_de_fichier" en se basant sur le PATH ainsi que sur des répertoires couramment utilisés.

ARCHIVAGE ET COMPRESSION

- **Compression** : Réduire la taille d'un fichier par algorithme de compression.
- **Archivage** : Placer un ensemble de fichiers et/ou de dossiers dans un seul fichier.
- Commandes :
  - Compression sans archivage : gzip/gunzip, bzip2/bunzip2
  - Archivage avec ou sans compression : tar

COMPRESSION GZIP/GUNZIP

- **gzip** est basé sur l'algorithme Deflate (combinaison des algorithmes LZ77 et Huffman).
- C'est la méthode de compression la plus populaire sous GNU/Linux.
- Démonstration :
  - `$ wget https://melot.fr/div/dico.txt`
  - `$ ls -lh`
  - `$ gzip dico.txt`
  - `$ ls -lh`

COMPRESSION GZIP/GUNZIP

- Pour décompresser un fichier gzippé :
  - **`$ gunzip dico.txt`**
  - **`$ ls -lh`**
- ou
  - **`$ gzip -d dico.txt`**
- Pour compresser plusieurs fichiers en un :
  - **`$ gzip -c fichier1 fichier2 > fichier_comp.gz`**

ARCHIVAGE TAR

- La commande **tar :** « tape archiver », ou en français « archiveur pour bande », son rôle à l'origine
- programme d'archivage de fichiers le plus populaire sous GNU/Linux et les systèmes Unix.
- Il est généralement installé par défaut.
- On peut ajouter à une archive tar différents algorithmes de compression.
- tar préserve les permissions et les propriétaires des fichiers, ainsi que les liens symboliques.

ARCHIVAGE TAR

- Pour archiver sans compression plusieurs fichiers ou un dossier, la commande est la même :
  - **`$ tar cvf mon_archive.tar fichier1 fichier2`**
  - **`$ tar cvf mon_archive.tar dossier1/`**
- Pour extraire une archive tar :
  - **`$ tar xvf mon_archive.tar`**

ARCHIVAGE TAR

- Les principales options de tar sont les suivantes et peuvent se combiner :
  - c / x : construit / extrait l'archive ;
  - v : mode bavard ;
  - f : utilise le fichier donné en paramètre.

ARCHIVAGE TAR

- Tar peut archiver en utilisant des algorithmes de compression, afin d'avoir des archives moins volumineuses.
- Par habitude, on suffixe les archives avec une extension de compression.
- Il suffit d'ajouter à la commande **tar** une option de compression :
  - z : compression Gzip
  - j : compression Bzip2
- Pour archiver et compresser un dossier avec Gzip :
  - **`$ tar cvzf mon_archive.tar.gz dossier1/`**
- Pour extraire une archive tar.gz :
  - **`$ tar xvzf mon_archive.tar.gz`**

ARCHIVAGE & SAUVEGARDE

- Il existe de nombreux programmes pour la gestion des archives et des sauvegardes :
  - Xz
  - Zip
  - **Rsync**
  - **Bacula**
  - …
- *Voir : apt search { backup, rsync, bacula }*

#6

Sécurité locale

Utilisateurs et groupes Linux

COMMANDE SU

- su (substitute user ou switch user) est une commande Unix permettant d'exécuter un interpréteur de commandes en changeant d'identifiant de GID et de UID.
- Sans argument, la commande utilise les UID 0 et le GID 0, c'est-à-dire ceux du compte utilisateur root.
- Cette commande est surtout utilisée pour obtenir les privilèges d'administration à partir d'une session d'utilisateur normal, c'est-à-dire, non privilégiée.
- L'option **–** ajoute l'environnement de l'utilisateur appelé (notion importante)
  - **`$ su`**
  - **`$ su -`**
  - **`$ su pascal`**
  - **`$ su - pascal`**

COMMANDE SUDO

- sudo (abréviation de substitute user do, soit : « exécuter en se substituant à l'utilisateur ») est une commande qui permet à l'administrateur système d'accorder à certains utilisateurs (ou groupes d'utilisateurs) la possibilité de lancer une commande en tant qu'administrateur, ou comme autre utilisateur, tout en conservant une trace des commandes saisies et des arguments.
- Pour configurer sudo (attention) :
  - **`# visudo`**
- Il peut être nécessaire d'installer préalablement sudo :
  - **`# apt install sudo`**

COMMANDE SUDO

- Pour voir la liste des utilisateurs :
  - **# $ cut -d: -f1 /etc/passwd**
- Pour voir la liste des groupes :
  - **# $ cut -d: -f1 /etc/group**
- Pour créer un utilisateur :
  - **# adduser michel**
- A jouter l'utilisateur au groupe sudo :
  - **# adduser michel sudo**

UTILISATEURS

- Toute entité (personne physique ou programme particulier) devant interagir avec un système UNIX est authentifiée sur cet ordinateur par un utilisateur ou "user".
- Un utilisateur est identifié par un nom unique et un numéro unique.
- Sur tout système UNIX, il y a **un super-utilisateur**, généralement appelé **root**, qui a tous les pouvoirs sur le système.
- L'utilisateur root porte le numéro 0 (uid)
- Il peut accéder librement à toutes les ressources de l'ordinateur, y compris à la place d'un autre utilisateur, c'est-à-dire sous son identité.
- Sur les systèmes de production, seul l'administrateur système possède le mot de passe root.

LE FICHIER /ETC/PASSWD

- On peut créer un utilisateur de plusieurs manières mais la finalité est toujours la même :
- pour chaque utilisateur, une entrée doit être créée dans le fichier /etc/passwd sous ce format :
  - **`account:passwd:UID:GID:GECOS:directory:shell`**
- Par exemple, on ajoute un utilisateur "user1" :
  - **`$ echo "user1:x:2000:2000:user1:/home/user1:/bin/bash" >> /etc/passwd`**
- Il faut ensuite créer le groupe correspondant, vérifier la validité des UID et GID, créer le répertoire utilisateur, y donner les droits et y placer une structure ...

LE FICHIER /ETC/SHADOW

- Les mots de passe sont stockés dans le fichier **/etc/shadow** avec ces paramètres :
  - nom de connexion de l'utilisateur (login)
  - chiffrement : 1 (MD5),2 (Blowfish), 5 (SHA-256), 6 (SHA-512)
  - date du dernier changement de mot de passe
  - âge minimum / maximum du mot de passe
  - période d'avertissement d'expiration du mot de passe
  - période d'inactivité du mot de passe
  - date de fin de validité du compte
- Par exemple :

```
pascal:$6$d/uLirbD$s90XRAj6g14036jIuvYYQaSOSrcJKqiNNywIQplztkTlyIry
SZE1o2zjFvSobewvyORXFdZ7bGeF0U1OTPoOm.:16842:0:99999:7:::
```

GROUPES

- Un utilisateur UNIX appartient à un ou plusieurs groupes.
- Les groupes servent à rassembler des utilisateurs afin de leur attribuer des droits communs.
- Le groupe principal est le groupe initial de l'utilisateur.
- L'utilisateur peut appartenir à des groupes secondaires.
- On peut vérifier son identifiant et l'appartenance aux groupes avec la commande **id** ou encore **groups**

FICHIERS **/ETC/GROUP** ET **/ETC/GSHADOW**

- Les fichiers **/etc/group** et **/etc/gshadow** définissent les groupes.
- Le fichier /etc/group comporte 4 champs :
  - nom du groupe
  - mot de passe du groupe (ou x si le fichier gshadow existe)
  - le GID
  - liste des membres séparés par une virgule

CRÉATION D'UN UTILISATEUR

- La commande **/usr/sbin/useradd** permet créer les nouveaux comptes utilisateurs.
- C'est le binaire de base du système.
- Une autre commande existe : /usr/sbin/**adduser**
- C'est (sous Debian) un script perl qui utilise useradd, en apportant quelques automatismes et interactivité.
- Syntaxe simplissime :
  - **`# adduser`** ***`login_utilisateur`***
- Commandes associées : usermod, userdel

RÉPERTOIRE SQUELETTE

- Le répertoire squelette **/etc/skel** contient les fichiers et répertoires qui seront copiés dans le répertoire personnel de l'utilisateur au moment de sa création.
- Contenu (variable selon les paramètres du système) :

```
$ ls -la /etc/skel/
total 24
drwxr-xr-x.   3 root root   74  8 déc 21:03 .
drwxr-xr-x. 122 root root 8192 16 jan 23:44 ..
-rw-r--r--.   1 root root   18 26 sep 03:53 .bash_logout
-rw-r--r--.   1 root root  193 26 sep 03:53 .bash_profile
-rw-r--r--.   1 root root  231 26 sep 03:53 .bashrc
drwxr-xr-x.   4 root root   37  4 jui  2014 .mozilla
```

GÉRER LES MOTS DE PASSE ET VERROUILLER

- C'est la commande **passwd** qui met à jour le mot de passe de l'utilisateur :
  - **`# passwd pablo`**
- Il est également possible de supprimer le mot de passe avec :
  - **`# passwd -d pablo`**
- Pour verrouiller un compte :
  - **`# passwd -l  pablo`** *ou* **`# usermod -L pablo`**
- Et pour déverrouiller :
  - **`# passwd -u pablo`** *ou* **`# usermod -U pablo`**

AJOUTER UN GROUPE

- On peut ajouter des groupes facilement avec groupadd ou **addgroup** :
  - **`# addgroup marketing`**
- On peut ajouter un utilisateur à un groupe avec **gpasswd** :
  - **`# gpasswd -a michel marketing`**
- On peut retirer un utilisateur d'un groupe :
  - **`# gpasswd -d milou marketing`**

MODIFIER LES PARAMÈTRES UTILISATEUR

- On change les paramètres des groupes avec le programme usermod. Par exemple :
  - **`# usermod -d /home/francois -G tintin,francois,wheel milou`**
- Les options de usermod sont (voir man usermod) :
  - -d : répertoire utilisateur
  - -g : définit le GID principal
  - -l : identifiant utilisateur
  - -u : UID utilisateur
  - -s : shell par défaut
  - -G : ajoute l'utilisateur à des groupes secondaires
  - -m : déplace le contenu du répertoire personnel vers le nouvel emplacement

MODIFIER LES PARAMÈTRES D'UN GROUPE

- C'est le programme **groupmod** qui permet de changer les paramètres d'un groupe.
- On utilise notamment les options suivantes :
  - -g GID
  - -n nom du groupe

MODIFIER L'EXPIRATION DU MOT DE PASSE

- La commande **chage** modifie le nombre de jours entre les changements de mot de passe et la date du dernier changement.
- Ces informations sont utilisées par le système pour déterminer si un utilisateur doit changer son mot de passe.
- Pour les lister les paramètres d'un utilisateur :
  - `# chage -l francois`
- Pour les détails :
  - `$ man chage`
- Notons que :
  - La date est soit en jours UNIX, soit au format YYYY/MM/DD.
  - Tous ces délais sont dans le fichier **/etc/shadow** et peuvent être modifiés manuellement.

SUPPRESSION D'UN COMPTE ET D'UN GROUPE

- On peut supprimer un compte utilisateur avec la commande userdel. Pour s'assurer de la suppression du répertoire utilisateur, utilisez l'option -r.
  - **`# userdel -r pablo`**
- Voir aussi : /usr/sbin/deluser, le script perl équivalent.

SÉCURITÉ LOCALE

PERMISSIONS LINUX

PROPRIÉTÉ

Tout fichier UNIX possède un propriétaire.

Au départ, le propriétaire est l'utilisateur (**u**) qui a créé le fichier mais "root" peut l'attribuer à un autre utilisateur.

Seul le propriétaire du fichier et le super utilisateur (root) peuvent changer les droits.

PROPRIÉTÉ

Un fichier UNIX appartient aussi à un groupe (**g**).

On définit ainsi les actions du groupe sur ce fichier.

Ce groupe est souvent le groupe d'appartenance du propriétaire, mais ce n'est pas obligatoire.

PROPRIÉTÉ

- On peut aussi définir ce que les autres (**o** : others) que le propriétaire ou groupe peuvent faire avec le fichier.
- Rappel : les dossiers sous UNIX sont aussi des fichiers.
- Les droits sur les dossiers (mais aussi les périphériques, etc.) fonctionnent exactement de la même façon que sur des fichiers ordinaires.

COMMANDES CHOWN/CHGRP

- **chown** est un appel système et une commande UNIX nécessitant les droits de root pour changer le propriétaire d'un fichier ou dossier
- De l'anglais « change the owner »
- Voici la syntaxe générale de la commande chown :
  - **`# chown [-hHLPR] [utilisateur][:groupe] cible1 [cible2 ..]`**

COMMANDES CHOWN/CHGRP

- **chgrp** permet de changer le groupe d'utilisateur possédant un fichier ou un dossier.
- Contrairement à **chown**, la commande n'est pas réservée au super-utilisateur
- le propriétaire peut aussi effectuer un changement de groupe s'il fait partie du groupe de destination.
  - `# chgrp [OPTION]… GROUPE CIBLE1 [CIBLES2 …]`

EXEMPLE

- Par exemple attribuer l'utilisateur **milou** et le groupe **users** au fichier monfichier.txt :

```
$ touch monfichier.txt
$ ls -l monfichier.txt
-rw-rw-r--. 1 pascal pascal 0 17 jan 12:37 monfichier.txt
$ sudo chown milou:users monfichier.txt
$ ls -l monfichier.txt
-rw-rw-r--. 1 milou users 0 17 jan 12:37 monfichier.txt
```

DROITS

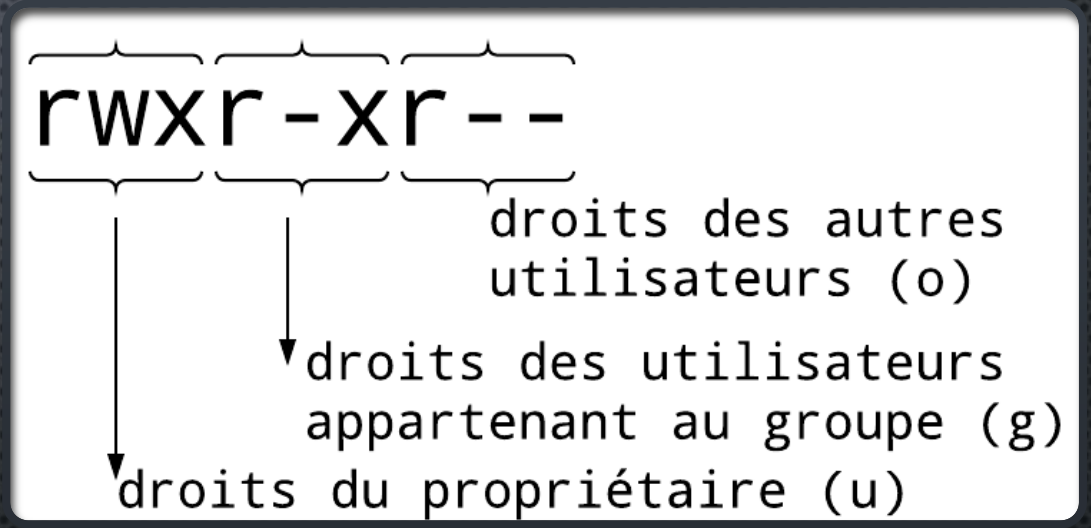
- À chaque fichier est associée une liste de permissions qui déterminent ce que chaque utilisateur a le droit de faire du fichier.
  - La lecture (r) : on peut par exemple lire le fichier avec un logiciel. Lorsque ce droit est alloué à un dossier, il autorise l'affichage de son contenu (liste des fichiers présents à la racine de ce dossier).
  - L'écriture (w) : on peut modifier le fichier et le vider de son contenu. Lorsque ce droit est alloué à un dossier, il autorise la création, la suppression et le changement de nom des fichiers qu'il contient (quels que soient les droits d'accès). Le droit spécial sticky bit permet de modifier ce comportement.
  - L'exécution (x) : on peut exécuter le fichier s'il est prévu pour, c'est-à-dire si c'est un fichier exécutable. Lorsque ce droit est attribué à un dossier, il autorise l'accès (ou ouverture) au dossier.

DROITS

- On appelle parfois **r**, **w** et **x** des « flags » ou « drapeaux ».
- Sur un fichier donné, ces 3 « flags » doivent être définis pour son propriétaire, son groupe, mais aussi les autres utilisateurs (différents du propriétaire et n'appartenant pas au groupe).

REPRÉSENTATION SYMBOLIQUE

- Cet ensemble de 3 droits sur 3 entités se représente généralement de la façon suivante :
- On écrit côte à côte les droits **r**, **w** puis **x** respectivement pour le propriétaire (**u**), le groupe (**g**) et les autres utilisateurs (**o**).



REPRÉSENTATION OCTALE

- On utilise aussi des chiffres en octal (base 8) pour représenter les combinaisons r, w, x :
  - `7 rwx`
  - `6 rw-`
  - `5 r-x`
  - `4 r--`
  - `3 -wx`
  - `2 -w-`
  - `1 --x`

Symbole	Octal	Binaire
r	4	100
w	2	010
x	1	001

UMASK

- Les permissions standards sont :
  - 666 pour les fichiers
  - 777 pour les dossiers
- **umask** est le masque de création de fichier qu'il faut soustraire des permissions standards pour obtenir les droits de tout nouveau fichier ou dossier créé par l'utilisateur.
- Obtenir la valeur actuel de umask :

```
$ umask
```

- Il peut être modifié globalement dans le fichier **/etc/login.defs** ou localement dans le fichier **~/.profile**

UMASK

- Exemple, avec umask positionné à 027, les fichiers nouvellement créés auront des droits :

```
 666
-
 027
=
 640 soit rw-r-----
```

- et les dossiers auront des droits :

```
 777
-
 027
=
 750 soit rwxr-x---
```

CHMOD

- **chmod** est la commande qui permet de changer les permissions des fichiers et des dossiers.
- Voici sa syntaxe :
  - **`chmod [option] permission fichier`**
- où les permissions peuvent être notées en notation octale :
  - **`$ chmod 777 fichier`**
- Avec donc 777 correspondant à : rwxrwxrwx

CHMOD

- ou en mode symbolique selon la syntaxe en utilisant :
  - les catégories d'utilisateur : u, g, o et a (all)
  - des opérateurs d'ajout/suppression : =, + et -
  - des droits : r, w et/ou x
- Par exemple :
  - **`$ chmod a+rwx fichier`**
  - **`$ chmod u+x monscript.sh`**
- Pour assurer la récursivité, on peut appliquer les permissions à un dossier et toute son arborescence avec l'option **`-R`** :
  - **`$ chmod -R u+rwx labs`**

DROITS ÉTENDUS

- Les droits étendus sont des variantes sur l'exécution :
  - SUID sur un exécutable, valeur octale : 4000, valeur symbolique : s
  - SGID sur un fichier ou un dossier, Valeur octale : 2000, valeur symbolique : s
  - Sticky bit, Valeur octale : 1000, valeur symbolique : t

SUID

- SUID : valeur octale : 4000, valeur symbolique : s
- Quand le SUID est activé sur un exécutable, l'utilisateur qui exécute le fichier dispose des mêmes droits que le propriétaire.
- Par exemple :

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 10 jun  2014 /usr/bin/passwd
```

- Sur cet exécutable le SUID est activé
- Ceci autorise un utilisateur d'écrire son mot de passe dans un fichier appartenant à root (/etc/shadow).
- *Attention ce type de fichier appartenant à root pourrait rendre n'importe quelles actions privilégiée possible.*

COMMANDE STAT

- La commande **stat** donnera des informations détaillées sur un fichier :

```
$ stat $(which passwd)
  Fichier : /usr/bin/passwd
   Taille : 27832        Blocs: 56         Blocs d'E/S: 4096   fichier
Périphérique: fd00h/64768d  Inœud: 33859624    Liens: 1
Accès: (4755/-rwsr-xr-x)  UID: (    0/    root)   GID: (    0/    root)
Contexte: unconfined_u:object_r:passwd_exec_t:s0
 Accès: 2019-01-16 02:52:08.012260715 +0100
Modif.: 2018-06-10 08:27:56.000000000 +0200
Changt: 2018-12-08 20:37:41.265606127 +0100
  Créé: -
```

OPÉRATIONS SUID

- Créer un dossier et tenter de changer son propriétaire. La commande **chown** ne peut être utilisée que par root.

```
$ mkdir tmp
$ chown root tmp
chown: modification du propriétaire de «tmp»: Opération non permise
$ ls -l $(which chown)
-rwxr-xr-x. 1 root root 62792 10 jun  2014 /usr/bin/chown
```

- On active le SUID sur la commande.

```
$ sudo chmod +4000 $(which chown)
$ ls -l $(which chown)
-rwsr-xr-x. 1 root root 62792 10 jun  2014 /usr/bin/chown
```

OPÉRATIONS SUID

- On constate que l'utilisateur peut désormais changer le propriétaire du dossier.

```
$ chown root tmp
$ ls -ld tmp
drwxrwxr-x. 2 root pascal 6  4 mar 23:10 tmp
```

- Désactivation du SUID :

```
$ sudo chmod u-s $(which chown)
```

SGID

- Valeur octale : 2000, valeur symbolique : s
- Le SGID permet d'endosser les droits du groupe propriétaire.
- Quand un utilisateur crée un fichier dans un dossier dont il est membre du groupe, le fichier prendra les permissions du groupe courant.
- Quand le SGID est fixé sur un dossier, le fichier créé par l'utilisateur prendra les droits du groupe du dossier.
- En conséquence, tous les fichiers créés quel que soit l'utilisateur appartiendront au groupe du dossier.

OPÉRATIONS SGID

- Créer un dossier partagé avec l'utilisateur tintin avec le SGID activé :

```
$ mkdir share
$ chmod g+s share
$ ls -l
drwxrwsr-x. 2 pascal pascal      6 17 jan 11:53 share
$ groups tintin
$ sudo usermod -G pascal tintin
$ groups tintin
$ cd share
```

OPÉRATIONS SGID

- Création d'un fichier partagé tintin.txt

```
$ su tintin
Mot de passe :
$ touch tintin.txt
$ ls -l
total 0
-rw-rw-r--. 1 tintin francois 0 17 jan 11:57 tintin.txt
$ exit
```

OPÉRATIONS SGID

- On retire le droit SGID et on crée un nouveau fichier tintin2.txt

```
$ chmod g-s share
$ cd share
$ su tintin
Mot de passe :
$ touch tintin2.txt
$ ls -l
total 0
-rw-rw-r--. 1 tintin tintin   0 17 jan 11:59 tintin2.txt
$ exit
```

STICKY BIT

- Valeur octale : 1000, valeur symbolique : t
- Ce droit (traduction bit collant) est utilisé pour manier de façon plus subtile les droits d'écriture d'un dossier. En effet, le droit d'écriture signifie que l'on peut créer, modifier et supprimer les fichiers de ce dossier.
- Le sticky bit permet de faire la différence avec la suppression.
- Lorsque ce droit est positionné sur un dossier, il interdit la suppression d'un fichier qu'il contient à tout utilisateur autre que le propriétaire du fichier.
- C'est typiquement le cas du dossier /tmp :

```
$ ls -l /
drwxrwxrwt.  11 root root  340 17 jan 11:59 tmp
```

#7