ASUR4
Administration et Sécurisation des Services

Manuel Munier

Département RT
IUT des Pays de l'Adour
manuel.munier@univ-pau.fr

Plan du cours

Plan du cours

Ce que dit la maquette de la LP RT ASUR

- **L**icence **P**rofessionnelle **R**éseaux et **T**élécommunications option **A**dministration et **S**éc**U**rité des **R**éseaux
- 4 modules ASUR :
  1. Analyse de risques
     objectifs de sécurité, analyse de risques, EBIOS,...
  2. Architectures de réseaux sécurisées
     notions crypto, pare-feu, portails captifs, modèle AAA, sécu Wifi,...
  3. Virtual Private Networks (VPN)
     IPSec, MPLS,...
  4. Administration et Sécurisation des Services
     ce module :-)

Ce que dit la maquette de la LP RT ASUR

- Contenu ASUR4
  - Administration des OS serveurs (2000, 2003, GPO, AD,...)
  - Administration Linux
  - Langage script (Perl,...)
  - Prise en compte des aspects sécurité (confidentialité, intégrité, disponibilité) dans le choix et la mise en place des équipements réseau
- Volume horaire 1ère partie : 34h30
  - → Cours 4h30 / TD 12h / TP 18h
    (y compris exam & éval TP...)
- Qq mots clés
  - admin système, archi Linux, scripting, sécurité,...

Plan du cours

Techniques d'administration

- Administration de réseaux, gestion du parc matériel, télé-collecte
- Gestion des logiciels et des licences, télédistribution
- Maintenance, surveillance, télé-diagnostique, télémaintenance
- Métrologie et sécurité, intégrité des données, sauvegardes

Capacités attendues

- Installer le système d'exploitation d'un réseau
- Administrer un réseau local
- Assurer la sécurité d'un réseau local
- Assurer la gestion d'un parc matériel et logiciel
- Assurer la surveillance et la maintenance d'un réseau local

Plan du cours

Notion de service

- Exemple : une application veut consulter les associations adresse réseau/nom de machine
- Architecture basique (poste isolé)
  - tous les couples adresse/nom sont stockés "en dur" dans un fichier sur chaque poste (fichiers `lmhosts` et `hosts`)

  $\Rightarrow$ problèmes d'administration
    - s'il y a beaucoup de machines
    - si les changements sont fréquents

Notion de service

- Architecture client/serveur
  - les couples adresse/nom sont gérés par un programme spécifique : le **serveur** de noms
  - quand une application (le **client**) doit résoudre un nom
    - elle se connecte au serveur
    - elle lui envoie le nom de la machine
    - le serveur consulte sa liste (et fait éventuellement des requêtes à d'autres serveurs)
    - il renvoie au client l'adresse réseau de la machine

Notion de service

⇒ Cette architecture a plusieurs avantages :

- le client et le serveur peuvent s'exécuter sur des machines différentes mises en réseau
- un même serveur peut héberger différents services et servir plusieurs clients répartis sur le réseau
  - serveur mail, web, fichiers, applications réseau (ex : agenda partagé, base de données commune,...)
- administration centralisée
  - facilité : modifications sur le serveur uniquement
  - sécurité : accès restreints

Notion de service

- Problème : "quelqu'un" doit lancer le serveur avant les connexions des clients

⇒ D'où la notion de service (ou démon) : c'est un serveur qui est lancé

  - soit au démarrage du système d'exploitation
  - soit à la demande par un super-démon quand une requête arrive pour ce serveur

⇒ Ex : Windows NT/XP/...édition Server, tous les Unix,...

Architecture client/serveur

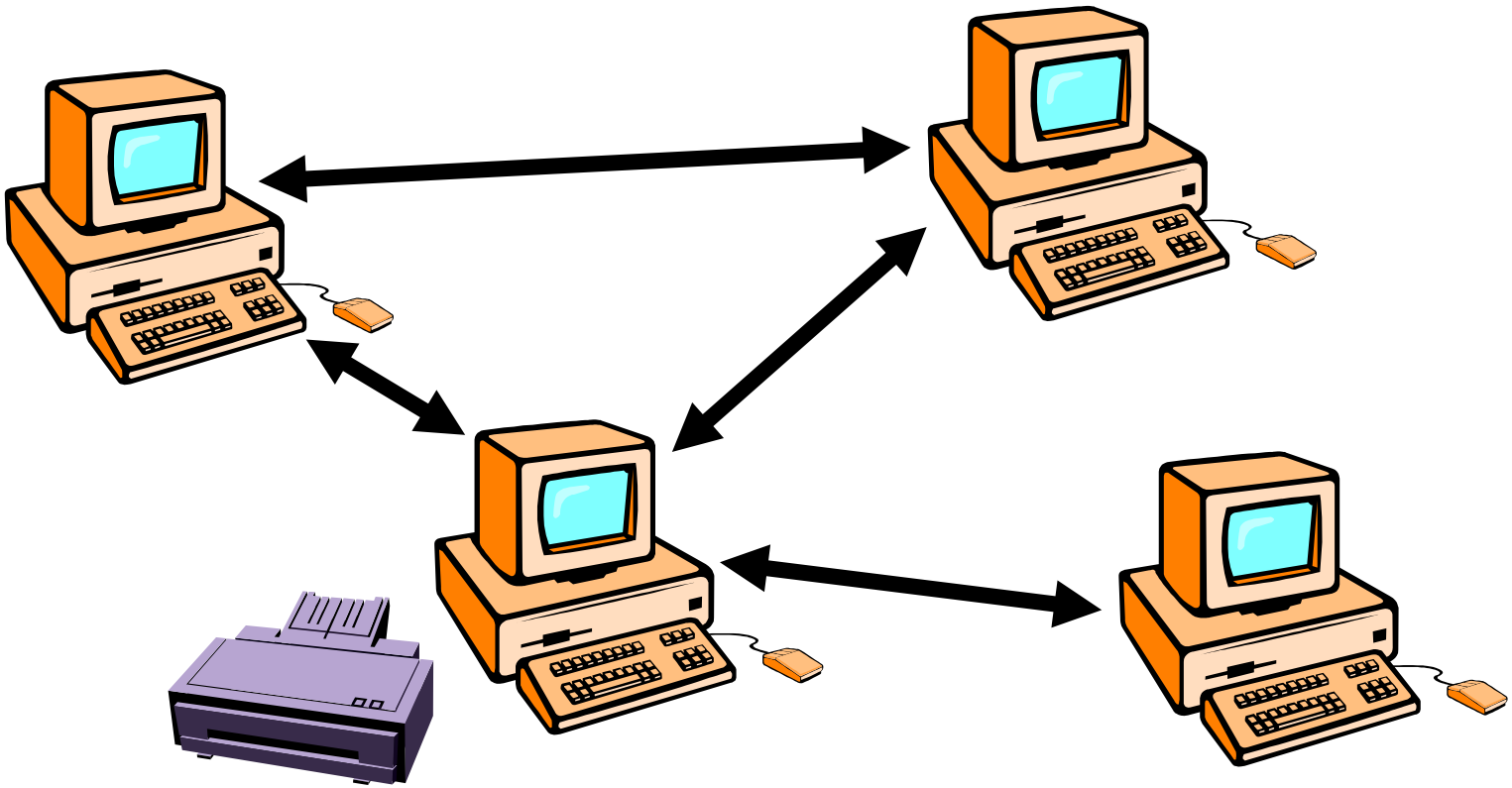
- Exemple : Microsoft Windows Workgroups
- C'est un réseau poste à poste
  - les ordinateurs connectés forment un groupe de travail
  - dans un groupe de travail, les ordinateurs peuvent utiliser mutuellement leurs ressources
  - chaque ordinateur décide seul de partager ou non ses ressources

Architecture client/serveur

- Réseau poste à poste (suite)
  - les droits d'accès aux fichiers et dossiers partagés sont élémentaires (à la connexion)
  - pas de différence entre *"utiliser"* une ressource et la *"mettre à disposition"*
  - un groupe de travail ne peut pas empêcher un membre d'un autre groupe de travail de se connecter et d'utiliser les ressources partagées
  - aucun contrôle global des droits d'accès et des utilisations du réseau

Architecture client/serveur

- Réseau poste à poste (ex : Workgroup)



Architecture client/serveur

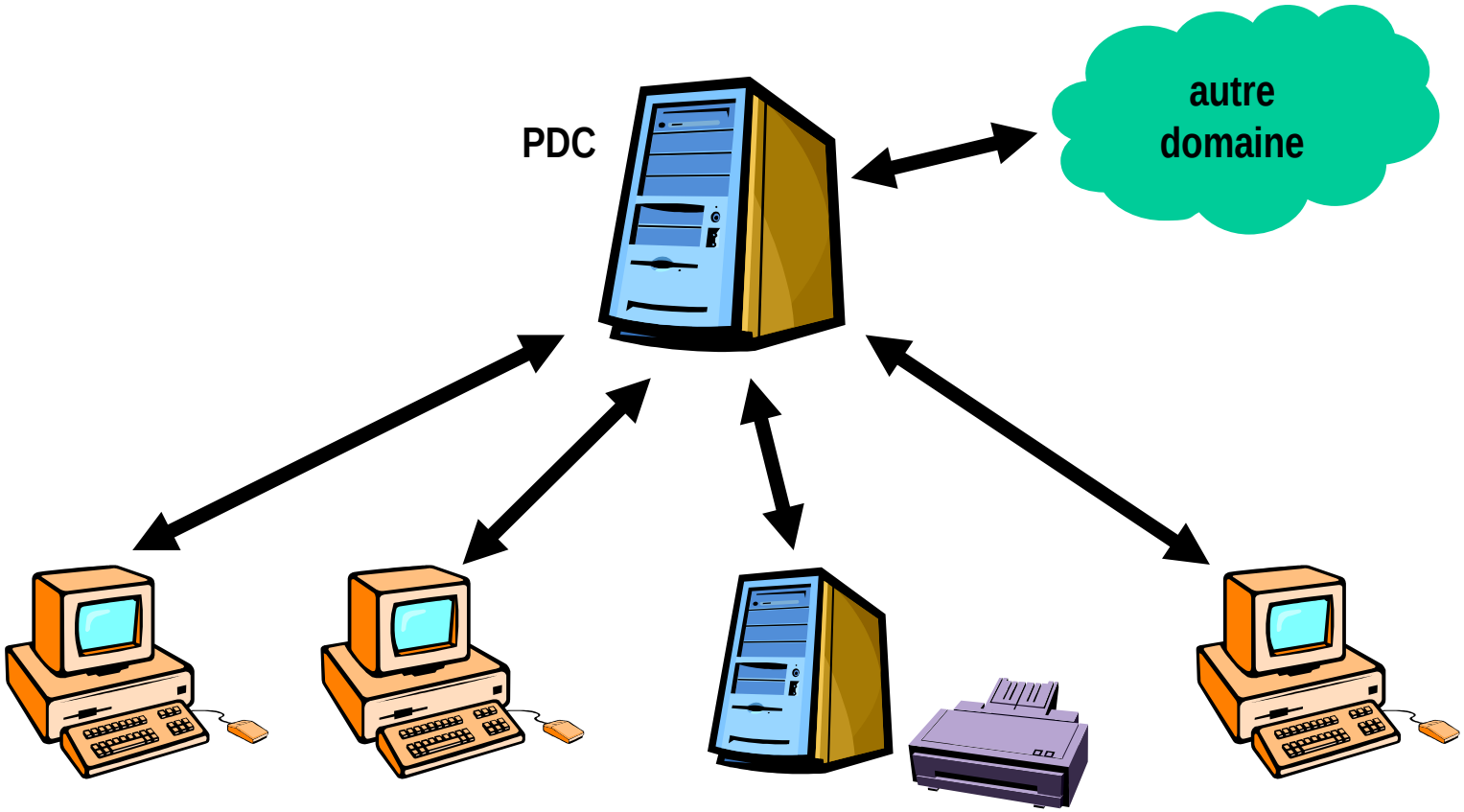
- Les réseaux poste à poste sont limités
  - aux réseaux complètement fermés,
  - composés d'un petit nombre de machines,
  - où toutes les données partagées peuvent effectivement être utilisées par tous les membres du réseau
- S'il y a des besoins de sécurité, restrictions d'accès,...
  $\Rightarrow$ réseaux client/serveur

Architecture client/serveur

- Un groupe de travail n'est qu'une liste de machines partageant certaines ressources
- Dans un réseau client/serveur, un domaine désigne un groupe de machines dirigées par une politique de sécurité commune
  - un serveur contrôle les droits d'accès
  - un ordinateur ne peut accéder aux ressources du groupe qu'après validation par le serveur

Architecture client/serveur

- Réseau client/serveur (ex : NT)



Architecture client/serveur

- Le serveur du domaine gère également :
  - l'accès à d'autres serveurs du domaine (base de données, agenda partagé, applications,...)
  - les connexions avec d'autres réseaux (domaines NT ou autres)
- Pas de "serveur de domaine" sous Unix, mais plusieurs serveurs avec des rôles bien définis : NIS, DNS, DHCP, NFS, firewall,...

Architecture client/serveur

- Outre les aspects sécurité, la centralisation des informations sur le serveur facilite l'administration du réseau
- Pb : si un serveur tombe en panne, tous les services qu'il fournissait sont indisponibles

⇒ **Sécurité** : accès physique, accès logiciel, sauvegardes, charge, maintenance, supervision,. . .

Notions fondamentales

Services & architecture C/S

Les notions de service et d'architecture client/serveur sont à la base de l'administration d'un réseau

- → Gestion des utilisateurs
- → Gestion des ressources
- → Sécurité (contrôle d'accès)
- → Intégrité (contrôle des accès concurrents)
- → Sauvegardes
- → Supervision

Plan du cours

2 Administration d'un réseau
- Rappels : service, client/serveur
- Installation
- Supervision
- Sécurité

Installation

- L'administration d'un réseau commence dès la phase d'installation et de configuration
  - ▷ du matériel
  - ▷ du système & des logiciels
  - ▷ des services réseau

Installation du matériel

- Éviter la diversité du matériel
  - ⇝ même configuration matérielle/logicielle sur toutes les machines similaires
- Choisir des marques connues
  - ⇝ problèmes/conflits répertoriés → Internet
  - ⇝ drivers mis à jour régulièrement
- Attention à la garantie !
  - ⇝ garantie 3 ans
  - ⇝ intervention sur site

Installation du matériel

- Pour les serveurs
  - estimer la charge (⇒ analyse des besoins)
  - anticiper les évolutions
    - ⇝ processeur(s), disques, mémoire,...
  - penser (dès le départ) à la sécurité
    - ⇝ hébergement, local, baies, climatisation,...
    - ⇝ onduleur, alimentations redondantes
    - ⇝ disques RAID, NAS,...
    - ⇝ système de sauvegarde
    - ⇝ serveur physique, virtualisé, dans le cloud ?

Installation du système d'exploitation

- Éviter la diversité des systèmes
  - Windows's, Unix's, Mac OS's, Android's,...
  - → uniformiser l'administration
  - → assurer et pérenniser les compétences pour le suivi
- Protocoles réseau
  - NetBEUI, IPX/SPX, TCP/IP
  - Réseau local hétérogène et/ou ouvert sur Internet ⇒ TCP/IP

Installation du système d'exploitation

- Partitions multiples
  - mettre les fichiers des utilisateurs dans des partitions séparées
  - mettre les fichiers des composants système dans des partitions séparées
  - essayer de mettre les partitions système en lecture seule (pb : mise à jour difficile)

- Objectifs :
  - → éviter de saturer la partition /
  - → le système et les applications peuvent être réinstallés
  - → ce qui est important ce sont les données utilisateurs
  - ⇒ **identifier facilement ce qui doit être sauvegardé**

Installation du système d'exploitation

- Pour les postes de travail
  - même image disque pour tous les postes ⇒ outils de clonage
    - → même configuration matérielle
    - → configuration réseau téléchargée via DHCP
  - stations diskless
    - NT4 : remote booting de stations Win9x
    - Windows Terminal Server
    - BOOTP sous Unix (terminaux X)
    - machines virtuelles (avec image sur le réseau)
    - cloud ⇝ bureau virtualisé, applications virtualisées

Sécurité

- Protection physique
  - redémarrer la machine avec une ~~disquette~~ clé USB, un CD-ROM ou un disque externe
  - ouvrir la machine, enlever la batterie du BIOS
  - utilisation des mots de passe par défaut pour accéder au BIOS
  - redémarrer le système en passant des paramètres à LILO/GRUB

Sécurité

- Protection physique (suite)
  - installer un système physique tel que KeyGhost ("KeyLogger" matériel)
  - démonter le disque système (ou données)
  - débrancher le serveur, on/off plusieurs fois pour endommager le système de fichier $\rightsquigarrow$ restauration du système (mode mono-utilisateur)

  ⋮

Sécurité

- Solutions
  - Matériel
    - boîtier avec serrure et/ou cadenas
  - BIOS
    - mot de passe
    - optimiser les paramètres
    - boot seulement sur le disque dur
  - LILO (idem pour GRUB)
    - mettre un mot de passe dans `lilo.conf`
    - `chattr +i /etc/lilo.conf` ⇒ empêche les modifications
    - `chattr -i` permet les mises à jour

Sécurité

- Solutions (suite)
  - `Ctrl+Alt+Del`
    - désactiver via le fichier `/etc/inittab`
    - surveiller avec un script
  - logiciels d'administration à distance
    - ⇝ pour limiter les accès physiques au serveur
    - `sudo` via SSH
    - interface web : Webmin, Zentyal, Ajenti,...
  - sauvegardes
    - ⇝ nécessité de faire des sauvegardes, au cas où il faille réinstaller rapidement le système...

Sécurité

- Authentification
  - utiliser des shadow passwords
  - Pluggeable Authentication Modules (PAM)
    - ensemble de modules pour l'authentification (*account*, *auth*, *password* et *session*)
    - permet d'utiliser un autre algo que DES pour chiffrer les mdp
  - quotas → limiter les ressources des utilisateurs (mémoire, nombre de processus,. . .)
  - systèmes avec carte à puce, empreinte digitale,. . .
  - allouer des horaires d'utilisation,. . .

Sécurité

- Pb : tout repose sur des mots de passe ! ⇒ robustesse ?
  - tester les mots de passe des utilisateurs
  - John the ripper, crack, vcu
  - les sauvegarder
  - obliger les utilisateurs à les changer régulièrement
  - pour les accès distants :
    - → SSH
    - → OTP (One Time Password)

Logiciels

- Uniformiser les logiciels
  - même suite bureautique (et même version) pour tout le monde
  - même navigateur web pour tout le monde
- Gérer les licences
  - une par poste (⇒ maintenir la liste)
  - licence site
  - serveur de licences (avec jetons)
- Installation des applications
  - locale (sur chaque poste)
  - centralisée : client/serveur, terminal server,...
  - virtualisée : intranet, cloud,...

Logiciels

- Outils de "gestion de parc"
  - inventaires des logiciels (et des matériels)
  - exemples :
    - Microsoft SMS (System Management Server)
    - GLPI (Gestion Libre de Parc Informatique)

  → distribution et installation de logiciels (via les scripts d'ouverture de session ou le réveil des postes)
  → administration à distance des clients
  → analyse et surveillance du réseau

Services réseau

- Dilemme
  - facilité d'administration
    - ⇝ ne pas multiplier les serveurs
    - ⇝ centralisation ⇒ tout est synchronisé
  - tolérance aux pannes
    - ⇝ distribuer les services sur plusieurs serveurs
    - ⇝ équilibrer les charges
    - ⇝ serveurs secondaires en secours
- Et l'arrivée du cloud n'arrange pas les choses ! :-)

Services réseau

- Objectifs de l'administrateur
  - → identifier et sécuriser les serveurs "officiels"
    - vérifier les fichiers de configuration
    - utiliser des technos "à jour"
    - se tenir informé des alertes de sécurité
  - → faire la chasse aux serveurs "pirates"
    - répertoires partagés
    - serveurs FTP, HTTP (web), SMTP (mail)
  - → filtrer/contrôler le trafic réseau
    - routeurs
    - firewalls
    - VLAN's

Services réseau

- Qq services classiques
  - DNS (Domain Name Service)
  - DHCP (Dynamic Protocol Service)
  - NIS (Network Information Service)
    - → centralisation sous Unix des fichiers `passwd`, `group`, mais aussi `host`, `services`, `protocols`,...
- Stockage des fichiers
  - répertoires/volumes partagés (hors partages Windows...)
    - NFS (Network File System) sous Unix
    - Samba : passerelle Windows ↔ Unix
    - iSCSI (Internet Small Computer System Interface)

  ⇒ avantages :
    - → fichiers accessibles depuis n'importe quel poste du réseau
    - → sauvegardes facilitées, gestion des quotas
    - → investissements centralisés (disques, RAID,...)

Virtualisation

- Côté serveurs
  - → installer plusieurs serveurs sur la même machine physique
  - → serveurs redondants (clusters,. . .)
  - → *load balancing*, *scalability*, *elasticity*
  - → investissements centralisés
  - → limiter les problèmes liés aux matériels différents
- Côté clients
  - → pouvoir exécuter plusieurs OS simultanément
  - → limiter le nombre de machines (tests,. . .)
  - → possibilité de repartir rapidement sur une image "propre" (éventuellement téléchargée sur le réseau)

Plan du cours

Titre

- Supervision...
  - Surveillance, diagnostic, maintenance

Titre

- Sécurité. . .
  - Politique de sécurité, surveillance
  - Sauvegardes

Plan du cours

Administration système

- Les actions d'un administrateur système :
  - installer un système UNIX
  - installer et configurer des services
  - administrer l'accès aux services
  - ajouter et enlever des utilisateurs
  - superviser des services
- Un administrateur système doit :
  - connaître le fonctionnement d'un système UNIX
  - connaître les commandes UNIX de base
  - savoir écrire des scripts shell
- Compte administrateur → `root`

Caractéristiques d'UNIX

- Deux familles : AT&T (Système V) et BSD
- Différents UNIX : HP-UX (HP), Solaris (SUN), AIX (IBM), UNIX SCO (SCO), LINUX, IRIX, Tru64, Unixware,...
- Multi-utilisateurs
- Multi-tâches
- Arborescence de fichiers :
  - système de fichier hiérarchisé : `/`, `/usr`, `/home`, `/var`,...
  - **"tout est fichier"**
- Noyau chargé au démarrage et processus utilisant API du noyau
- Shell associé à une session de travail ; utilisation de scripts
- Services et daemon (Disk And Execution MONitor)

Arrêt et redémarrage

- Séquence : chargeur de boot → noyau → `init` → processus
- Le chargeur de boot copie le noyau d'un OS sur la pile d'exécution (LilO, GRUB, Syslinux)
- Le noyau constitue le cœur du système d'exploitation
  - gestion de l'accès au matériel
  - gestion de la mémoire et des processus
- Activation des processus par `init`
  - BSD : `/etc/rc`
  - Système V : `/etc/inittab`
- Un système UNIX lance des services et dispose d'un gestionnaire de services
- Arrêt du système :
  - AT&T : `shutdown [-y] [-g delai] [-i niveau] [message]`
  - BSD : `shutdown [options] time [message]`

Le système de fichier

- Arborescence de fichiers qui masque l'emplacement des ressources (partitions sur différents disques ou sur le réseau)
- Les partitions sont montées sur le système de fichier
- Différents types de systèmes de fichiers :
  - S5 : FS le plus ancien
  - Fast File System (FFS) des systèmes BSD
  - Unix File System (UFS) des systèmes AT&T
  - Veritas File System (VXFS) : journalisation
  - ⋮
- Les systèmes de fichiers utilisent :
  - ▷ une table d'inode (table de descripteurs de fichiers)
  - ▷ données stockées dans des blocs
  - ▷ fichiers de type bloc, de type caractère

Le système de fichier

- Différents types de fichiers :
  - \- fichier ordinaire
  - d répertoire
  - c périphérique caractère
  - b périphérique bloc
  - l lien symbolique
- Nom des fichiers sensible à la casse : `nom` ≠ `nOm`
- Pas de restrictions dans les caractères (sauf /)
- Le nom peut être limité en longueur
- Fichiers et répertoires "cachés" commencent par un point
- Deux références par défaut dans un répertoire : `.` et `..`

Le système de fichier

- Quelques répertoires standards :
  - `/` ⇝ répertoire racine
  - `/bin` ⇝ exécutables de base (`ls`, `cp`, `mv`)
  - `/sbin` ⇝ exécutables système utilisés au démarrage
  - `/etc` ⇝ fichiers de configuration
  - `/tmp` ⇝ fichiers temporaires
  - `/var` ⇝ fichiers des bases de données, des serveurs, des logs
  - `/home` ⇝ répertoires des utilisateurs (ou `/users`)
  - `/usr` ⇝ hiérarchie secondaire avec le reste des applications
  - `/opt` ⇝ applications optionnelles
  - `/root` ⇝ répertoire de l'administrateur
  - `/boot` ⇝ noyau et fichier de boot
  - `/lib` ⇝ bibliothèques standards

Le système de fichier

- Monter un système de fichier : `mount /dev/hda1 /usr`
- Automatisation du montage dans `/etc/fstab`
- Gestion de l'espace disque : `df -k`
- Création d'un FS : `mkfs` ou `newfs`
- Vérification de cohérence : `fsck`
- Occupation d'un FS : `du`
- Quota : `edquota`, `quotaon`

Gestion des processus

- Un processus est défini par son PID (Process Identifier)
- Un processus peut être exécuté :
  - en avant-plan (*foreground*)
  - en arrière-plan (*background*)
  - en mode détaché (plus de terminal de contrôle)
  - en mode daemon (processus détachés associés à des services)
- Affichage des processus : `ps`
- Communication avec un processus par signaux :
  - → redémarrage : `kill -1` (`-HUP`)
  - → fin du processus : `kill -15` (`-TERM`)
  - → arrêt brutal : `kill -9` (`-KILL`)
- Commandes externes : `ps`, `kill`, `su`, `fuser`, `at`, `crontab`, `env`,...
- Commandes internes : `cmd &`, `wait`, `exec cmd`, `./cmd`,...

Gestion des services

- Daemon qui tournent en permanence :
  - gestion du matériel : `udev`
  - services réseaux : `dhcpd`,...
  - services applicatifs : `apache`, `mysql`,...
  - interface graphique : `X Window`, `Wayland`
  - interaction avec l'utilisateur : `getty`, `sshd`,...
- Gestionnaire de service
  - Debian/Ubuntu : `service` (via des scripts dans `/etc/init.d`)
  - ArchLinux : `systemctl`
- Gestionnaire de paquets
  - Debian/Ubuntu : `apt-get`
  - ArchLinux : `pacman`
  - SuSE/RedHat : `rpm`

X Window

- Framework pour le support d'interfaces graphiques sous UNIX
  - API pour l'utilisation de la souris, du clavier et de l'écran
  - interaction de l'utilisateur avec des fenêtres
  - accès distant (*remote display*)
- Composants :
  - serveur X : gère l'accès aux périphériques
  - *window manager* : Gnome-Shell, KWin, Enlightenment, Compiz, Beryl,...
  - *desktop environment* : KDE, Gnome,...
  - *session manager*
  - *display manager* : KDM, GDM,...
  - X11 : protocole de communication
- Configuration X11 :
  - ⇝ `/etc/X11/xorg.conf`
  - ⇝ `/var/log/Xorg.0.log`

Crontab

- Le daemon `cron` exécute des commandes pour les utilisateurs
  - ▷ à échéance : `at`
  - ▷ dès que possible ou à événement précis : `batch`
  - ▷ périodiquement : `crontab`
- Configuration des utilisateurs autorisés (ou pas) dans `cron.allow`, `cron.deny`, `at.allow`, `at.deny`
- Fichier `/etc/crontab` avec 6 colonnes :
  - minutes heures jour_du_mois mois jour_de_la_semaine commande
  - 30 8-19 * * 1-5 commande_1
  - 0 */4 * * * commande_2
- La cmde at lit des cmdes à exécuter depuis l'entrée standard
  - at heure [date] [+ incrément unité] < script
  - at 1500 < commande
  - at now + 3 hours < commande
  - at -l

Sécurité de la connexion

- Connexion au système via `login` ou `su`
- Mots de passe stockés dans `/etc/shadow` (`/etc/password` contient des infos sur les utilisateurs, parfois le mot de passe)
- Quelques règles pour le choix du mot de passe :
  - au moins 7 caractères et avec au moins une lettre majuscule, un chiffre et un caractère de ponctuation, et ceux-ci à l'intérieur et non en début ou fin de mot de passe
  - pas de données relatives à votre identité comme votre nom d'utilisateur ou une information livrée par la commande `finger`
  - ne pas appartenir à des dictionnaires (nom de plante, espèce animale), tel quel ou sous sa forme canonique (c'est-à-dire, épuré de tous les caractères non-alphabétiques), à moins qu'il ne contienne des majuscules autres que le premier caractère
  - pas des répétition de caractère
  - simple à mémoriser ($\rightsquigarrow$ ne pas le noter sur un morceau de papier !)

Sécurité de la connexion

- Quelques commandes utiles :
  - ▷ `passwd` : changement de mot de passe
  - ▷ `who` : affichage des utilisateurs connectés
  - ▷ `last` : affichage des dernières connections
  - ▷ `crack` : outil de vérification de la solidité des mots de passe

- Fichiers concernés :
  - → `/etc/login.defs`
  - → `/etc/password`
  - → `/etc/shadow`
  - → `/var/log/auth.log`
  - → `/var/log/sulog` (si activé...)

PAM

- PAM ≡ Pluggable Authentication Module
  - API d'authentification à disposition des applications
  - plus besoin de recompiler les applications
  - lien entre un mode d'authentification et un module
  - configuration dans `/etc/pam.d/`
- Syntaxe de `pam.conf`

  service type stratégie chemin argument
  login auth required /usr/lib/security/pam_unix.so
- Types disponibles :
  - `auth` : le module réalise l'authentification classique (login et mot de passe)
  - `account` : le module vérifie si l'authentification est autorisée
  - `password` : le module permet de vérifier les mots de passe
  - `session` : le module est activé quand l'utilisateur est authentifié et permet des contrôles d'accès supplémentaires

Plan du cours

Introduction

- Un shell assure l'interface entre l'utilisateur et le système
- C'est un interpréteur de commande(s)
- Caractéristiques :
  - déclenchement d'actions particulières par un jeu de caractères spéciaux
  - programmation par commandes internes et mots clés
  - paramétrage de l'environnement de travail
- Il existe différents shells sous UNIX :
  - `/usr/bin/sh` (shell POSIX ou Bourne Shell)
  - `/usr/bin/ksh` (Korn Shell)
  - `/usr/bin/bash` (Bourne Again Shell)
  - `/usr/bin/csh` (C shell)

Mécanismes essentiels

- Commande simple : séquence d'affectations de variables optionnelles suivie de mots séparés par des espaces et des redirections et terminée par un opérateur de contrôle
- Commandes externes ou internes : `type ls` vs `type pwd`
- Pipeline : séquence d'une ou plusieurs commandes séparées par l'un de opérateurs de contrôle `|` ou `|&` (équivalent à `2>&1 |`)
- Liste : séquence de un ou plusieurs pipelines séparés par l'un des opérateurs `;` `&` `&&` ou `||` et terminé optionnellement par `;` `&` ou `<retour-chariot>`

Mécanismes essentiels

- Exemples de séquences de commandes :
  - `commande_1 &` → commande_1 est exécutée en arrière-plan
  - `commande_1 && commande_2` → commande_2 est exécutée si et seulement si commande_1 retourne un statut nul
  - `commande_1 || commande_2` → commande_2 est exécutée si et seulement si commande_1 retourne un statut non nul
- Exemples de commandes internes :
  - affichage : `echo`
  - chargement d'un script : `source filename` (ou `. filename`)
  - envoi d'un signal : `kill -signum pid`
  - modification des variables shell : `set [-o options]`
- Caractère tilde `~` et séparateur de commande `;`

Mécanismes essentiels

- Expressions basiques :
  - `*` représente une suite de caractères
    ex : `ls *.c`
  - `?` représente un seul caractère quelconque
    ex : `ls ???.c`
  - `[ ]` spécifie une liste de caractères à une position précise
    ex : `ls [f]*.[a-f]` ou `ls [!a-z].c`

- Expressions complexes :
  - `?(expression)` : l'expression est présente 0 ou 1 fois
  - `*(expression)` : l'expression est présente 0 ou n fois
  - `+(expression)` : l'expression est présente 1 ou n fois
  - `@(expression)` : l'expr. est présente exactement une fois
  - `!(expression)` : l'expression n'est pas présente
  - `@(expr1|expr2|expr3)` : la barre | prend le sens de "ou bien"

Mécanismes essentiels

- Redirections des entrées-sorties
  - `/dev/stdin` : entrée standard (descripteur 0)
  - `/dev/stdout` : sortie standard (descripteur 1)
  - `/dev/stderr` : sortie d'erreur standard (descripteur 2)
- Redirections simples :
  - sortie standard : `ls 1> log` ou `ls > log`
  - sortie d'erreur : `find / -name toto 2> log.err`
  - double sortie : `find / -name toto 1> log.txt 2> log.err`
- Redirection double : `ls >> log`
- Redirection avancée : `find / -name toto 1> log.txt 2>&1`
- Fermeture d'un descripteur : `<&- >&- 2>&-`

Mécanismes essentiels

- Tubes de communications (ou pipe) par le caractère |

  ex : `dmesg | more` (ne concerne pas la sortie d'erreur standard)

- Entrée ignorée : `ls`, `who`, `find`, `chmod`, `cp`, `mv`, `rm`, `ln`, `mkdir`, `rmdir`, `date`, `kill`, `file`, `type`, `echo`

  ex : `cat /etc/passwd | file`

- Filtres : `grep`, `cat`, `sort`, `cut`, `wc`, `lp`, `sed`, `awk`

  ex : `cat /etc/passwd | grep bash`

- Regroupement de commandes
  - dans un shell enfant : `(cmd1; cmd2) > log.txt`
  - dans le shell courant : `{ cmd1; cmd2; } > log.txt`

Commandes filtres

- Visualisation des octets : `od -c fichier`
- Comptage : `wc [options] fichiers`
- Extraction : `cut [options] fichier`
- Transformation : `tr [options] ensemble1 ensemble2`
- Visualisation dernières lignes : `tail [-f|-n|+n] fichier`
- Visualisation premières lignes : `head [-n] fichier`
- Duplication de la sortie standard : `commande | tee [-a] fichier`
- Numérotation des lignes : `nl [options] fichier`
- Mise en page des données : `pr [options] fichier`

Commandes filtres

- `gzip [-ct] fichier` (pour décompresser : `gunzip` ou `zcat`)
  - `-c` envoie vers la sortie standard
  - `-t` test de la validité de l'archive
- `bzip2 [-cdt] fichier`
  - `-c` envoie vers la sortie standard
  - `-d` décompression
  - `-t` test de la validité
- `tar [options] -f fichier_archive fichiers_a_archiver`
  - `-c` création
  - `-x` extraction
  - `-t` vérification
  - `-v` mode verbeux
  - `-z` compression en plus de l'archivage (sous Linux)

Commandes filtres

- Consultation de données : `cat [-etv] [fichier]`
- Filtrage de lignes :
  - → `grep [options] regex [fichier]`
  - → `grep [options] -e regex1 -e regex2 [fichier]`
  - → `grep [options] -f fichier_regex [fichier]`
    - options :
      - -v lignes ne contenant pas le motif
      - -c nombre de lignes trouvées
      - -n numéroter les lignes trouvées
      - -x lignes correspondant exactement au motif
      - -w lignes ou le mot apparaît tel quel
      - -i insensible à la casse

Expressions régulières

- Expressions régulières basiques (ERb) et étendues (ERe)
- Utilisées dans `vi`, `grep`, `expr`, `sed` et `grep -E`, `awk`
- Caractères communs aux ERb et ERe :

Caractère	Signification
`^`	début de ligne ex : `grep ^Dec /var/log/messages`
`$`	fin de ligne ex : `grep enabled$ /var/log/messages`
`.`	un caractère quelconque
`[liste]`	un caractère dans la liste ; `[0-9][0-9][a-z]` ⇒ 12a 13b 14d
`[!liste]`	un caractère absent d'une liste
`*`	0 à n fois le caractère
`\<`	début d'un mot ; ex : `grep -n "\<term" /var/log/messages`
`\>`	fin d'un mot ; ex : `grep -n "\<terminus\>" /var/log/messages`
`\c`	protection du caractère c ; ex : `[0-9][0-9]\.[0-9]`

Expressions régulières basiques

- Expressions régulières basiques (ERb) :

Caractère	Signification
`\{n\}`	n fois le caractère précédent ; ex : `[0-9]\{5\}`
`\{n,\}`	au moins n fois le caractère précédent
`\{n,m\}`	entre n et m fois le caractère précédent ; ex : `[0-9]\{2,5\}`
`\(er1\)`	mémorise dans l'ERb 1
`\1 \2`	rappel des ERb 1 et 2

Expressions régulières étendues

- Expressions régulières étendues (ERe) :

Caractère	Signification
`?`	0 ou 1 fois le caractère précédent ; ex : `[0-9]?`
`+`	1 ou n fois le caractère précédent
`{n}`	n fois le caractère précédent ; ex : `[0-9]{5}`
`{n,}`	au moins n fois le caractère précédent
`{n,m}`	entre n et m fois le caractère précédent ; ex : `[0-9]{2,5}`
`(er1)`	mémorise dans l'ERb 1
`er1|er2|er3`	rappel des ERb 1, 2 et 3

Environnement de travail

- Blablabla...

Plan du cours

Titre

- Blablabla...

Plan du cours

Titre

- Blablabla...