Université de Bretagne Occidentale

Administration des systèmes d'exploitation

Gestion centralisée des postes de travail

Cours 1

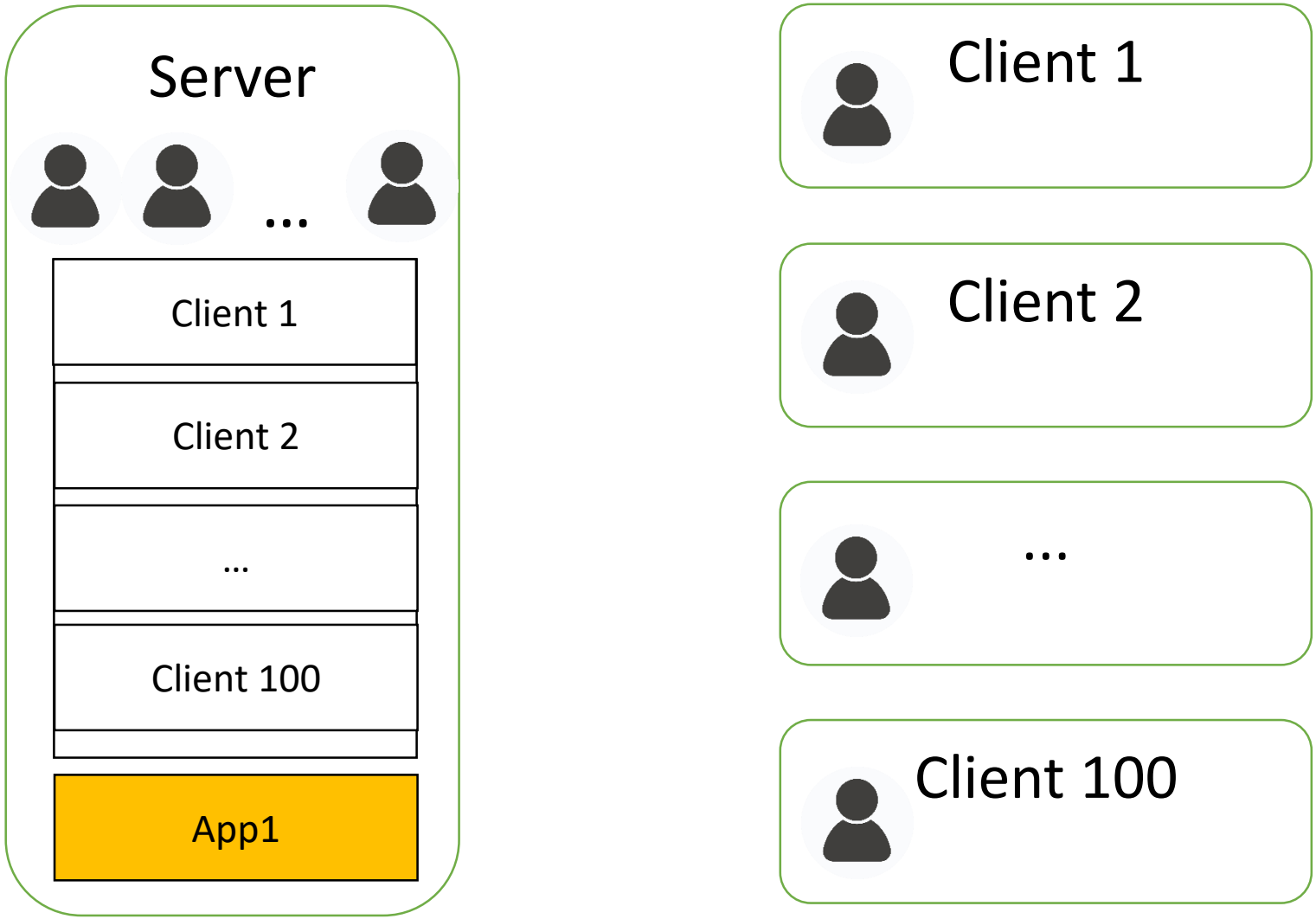
Hai Nam TRAN
Université de Bretagne Occidentale – M1 Informatique

Administration des systèmes d'exploitation

- **Partie : Gestion centralisée des postes de travail**
  - **6h de CM**
    - Présentation du projet
    - Introduction aux scripts bash
    - Accès distant aux ressources de stockage (NFS)
    - Infrastructure d'annuaires (NIS)
    - Firewall, Git/SVN, Apache
  - **12h de TP**
    - Projet : 101 (1 serveur + 100 clients)
    - Livrables : scripts de configurations, network file system (NFS), network information system (NIS), firewall, git/svn, apache
    - Évaluation du projet TP6 + compte rendu + devoirs
    - Graphe de dépendances :

- **Projet**
  - **Le but du projet est de configurer les postes d'un réseau de machines afin de mettre en place une gestion centralisée**
    - Les postes sont virtualisés
      - Oracle Virtual Box
      - OS Linux
    - La configuration des postes (**server** et **clients**) sera réalisée par l'intermédiaire de script **`bash`**
      - Connectivité réseau
      - Montage NFS statique
      - Gestion centralisée par NIS des logins, mots de passe et groupe
      - Firewall
      - Git/svn
      - Web serveur
      - …

Projet


Server
...
Client 1
Client 2
...
Client 100
App1
Client 1
Client 2
...
Client 100

Administration des systèmes d'exploitation

- **Projet : consignes**
  - **Possibilité de travailler en binôme**
  - **Dossier projet : ASE_Nom1_Nom2.zip**
    - scripts
    - compte_rendu.pdf
  - **À déposer avant la fin de chaque séance de TP sur le Moodle dans la zone de dépôt de votre groupe de TP**
- **Info**
  - **Email : hai-nam.tran@univ-brest.fr**
  - **Supports de cours :**
    - Moodle
    - phobos.univ-brest.fr

Introduction

Administrateur système - sysadmin

- **Assurer la fonctionnalité des infrastructures informatiques**
  - **Website**
  - **Réseau**
  - **Email**
  - **Base de données**
  - **Postes de travail**

  → Différentes technologies et humains...

Matériel	**Logiciel**
Réseau	**OS**

"Quand tout va bien, ils pensent que tu n'as rien à faire. Quand il y a un problème, c'est ta faute"

Administrateur système - sysadmin

- **Toute entreprise ayant une présence informatique a besoin d'un administrateur système**
  - **Une personne ou une équipe**
- **Un métier à haute responsabilité et stress**
  - **Le métier exige une disponibilité maximale pour garantir le bon fonctionnement du système**
- **System Administrator — Appreciation Day: 26 juillet**

Contexte et problématique

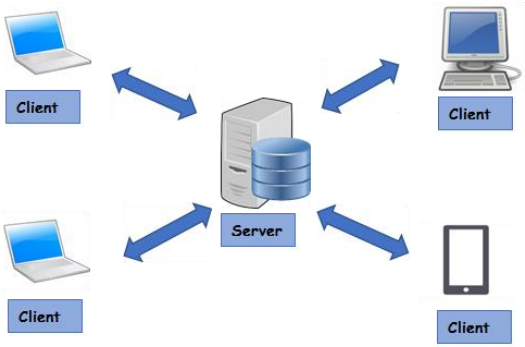
- **Environnement professionnel**
  - De nombreuses ressources partagées
  - Applicatives, de stockage, d'archivage
  - Les postes de travail eux-mêmes.
  - Besoin de solutions de collaboration

- **Problématique**
  - P1 : De nombreux postes de travail à administrer, configurer
  - P2 : Des systèmes évolutifs (nouvelles applications ou utilisateurs, mises à jour), et hétérogènes
  - P3 : Localisation sur des sites géographiques éloignes

- **Nécessité de centraliser la gestion et le partage des ressources informatiques**

Partage de ressources informatiques

- **Modèle client/serveur**
  - **Serveur de fichiers/d'exécutables**
    - Partage de données
    - Diffusion des exécutables des applications partagées
  - **Serveur de configurations**
    - Diffusion des paramètres de configuration partagés
  - **Serveur d'applications**
    - Serveur de calcul
    - Service Web
    - Software-as-a-service (SaaS, cloud)

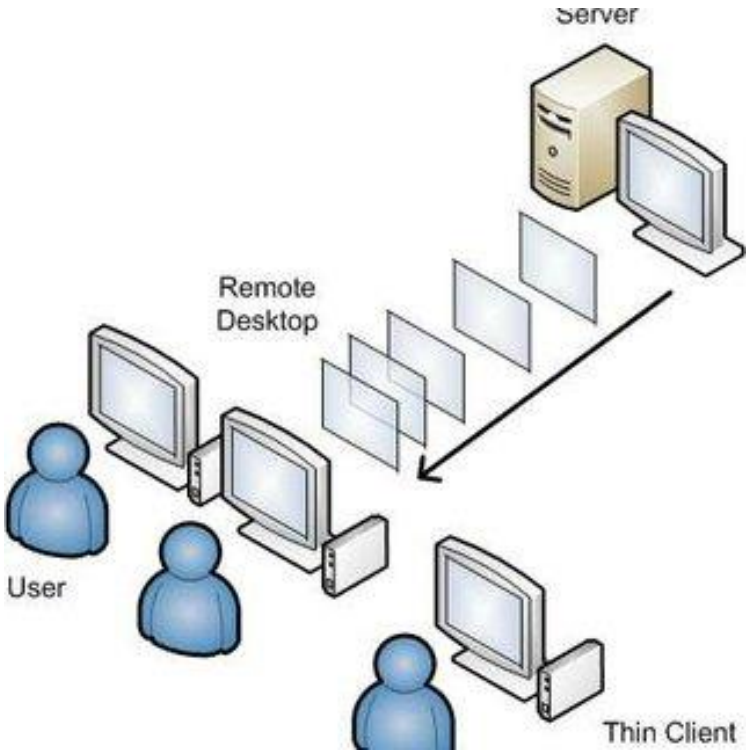


Partage de ressources informatiques

- **Modèle client/serveur**
  - **Intérêts**
    - Installation et mise à jours des applications au niveau des serveurs
    - Cohérence entre postes: OS, configurations de base, logiciels
    - Nomadisme (changement de poste)
  - **Inconvénients et limites**
    - Performances
    - Risque d'indisponibilité générale (pannes serveur ou réseau)
    - Gestion des configurations spécifiques
    - Prise en charge/mise à jour des OS des postes clients

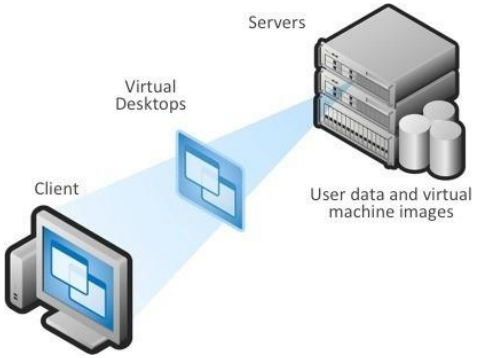
Virtualisation des postes de travail

- **Machine client = terminal**
  - **L'ordinateur passe du périmètre entreprise au périmètre data center (déploiement, maintenance, usage des ressources, consommation)**



Virtualisation des postes de travail

- **Clients légers**
  - **Simple, pas de disques dur ou PC « bas de gamme »**
  - **Travail à distance sur le poste**
  - **Déploiement plus simple des logiciels et des systèmes d'exploitation**
  - **Sécurité des postes de travail améliorée (?)**
  - **Mobilité, possibilité de se connecter à distance sur sa machine virtuelle, même en dehors des locaux de l'organisation**
  - **Equipe technique réduite**
    - Gestion uniquement d'un modèle de poste déployé par clonage au niveau du serveur



Virtualisation des postes de travail

- **Clients légers**
  - **Personnalisation par l'utilisateur limité**
  - **Périphériques liés aux clients légers difficiles à gérer (clef USB par exemple)**
  - **Dépendant du bon fonctionnement du réseau**
  - **Exemple :**
    - VMware Virtual Desktop Infrastructure (VDI)
    - Oracle Virtual Box

Tutorial Virtual Box

- **For the lab of this course we will use a pre-configured image of a virtual machine**
  - **OS : Ubuntu 18.04**
  - **Pre-installed packages : vim, nfs, nis, git, net-tools…**
- **Please follow the following steps precisely to**
  - **Create a virtual machine**
  - **Attach the virtual hard disk to the machine**
  - **Step 1: Start Oracle Virtual Box**
  - **Step 2: Click on the "New" button to create a virtual machine**
  - **Step 3: Provide the required information**
    - Name: server or client0x
    - Folder: a local folder on your PC. **DO NOT** store your machine on the vador-fs server
    - Type: Linux
    - Version: Ubuntu 18.04 or Ubuntu (64bit)

Tutorial Virtual Box

- **Step 4 : Hardware : Configure the hardware**
- **Step 5 : Virtual Hard disk**
  - At the step, you need to associate a virtual hard disk to your machine
  - This file **will be provided in the labs**

Plan

- Introduction
- Configuration - Introduction aux scripts `bash`
- Configuration - Réseau
- Accès distant aux ressources de stockage
- Infrastructure d’annuaires
  - Notion d'annuaires informatiques
  - Network Information System (NIS)
  - Domain Name Server (DNS)
  - OpenLDAP, Active Directory
  - Sélection du service d'annuaires

Reference : CSC3102 - Introduction aux systèmes d’exploitation. Élisabeth Brunet et Gaël Thomas

Problématique

- **De nombreux postes de travail à administrer, configurer**
  - **Impossible de configurer manuellement chaque poste de travail**
    - Il est facile d'effectuer une tâche sur une seule machine
    - Mais si vous souhaitez effectuer plusieurs fois la même tâche sur plusieurs machines, vous avez besoin de scripts
  - **Obligation d'automatiser un grand nombre de tâches**
    - Mettre à jour, nettoyage, redémarrage
    - 24/24, 7/7, 365/365 !
- **La connaissance du langage de script est également utile pour votre carrière en informatique !**

Projet : **TP1**

Il était une fois ...

- **"A programmer wrote scripts to secretly automate a lot of his job"**
  - **https://www.businessinsider.com/programmer-automates-his-job-2015-11?IR=T**
    - "The guy wrote one script that **sends a text message "late at work" to his wife** and "automatically picks reasons" from a preset list of them. It sent this text **anytime there was activity with his login on the company's computer servers after 9 p.m**"
    - "With another script, he **automatically fired off an email excuse** like "not feeling well, working from home" **if he wasn't at work and logged in to the servers by 8:45 a.m**"
    - "He wrote a script that waits 17 seconds, then **hacks into the coffee machine and orders it to start brewing a latte**. The script tells the machine to wait another **24 seconds** before pouring the latte into a cup, **the exact time it takes to walk from the guy's desk to the coffee machine**"
      - Co-worker : "We do not even know the machine is programmable !"

https://github.com/NARKOZ/hacker-scripts

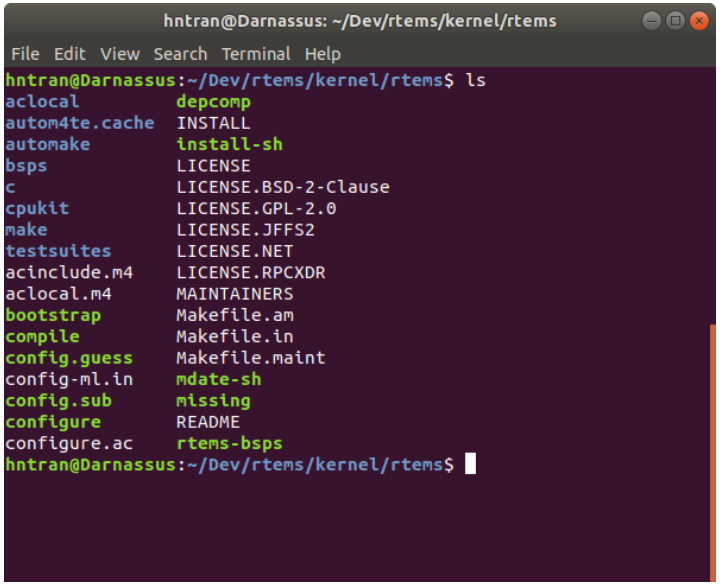
Il était une fois ...

```
# Runs `hangover.sh` monday to friday at 8:45 am.
45 8 * * 1-5 /path/to/scripts/hangover.sh >> /path/to/hangover.log 2>&1
```

```
3   # Exit early if any session with my username is found
4   if who | grep -wq $USER; then
5     exit
6   fi
7
8   # Phone numbers
9   MY_NUMBER='+xxx'
10  NUMBER_OF_BOSS='+xxx'
11
12  EXCUSES=(
13    'Locked out'
14    'Pipes broke'
15    'Food poisoning'
16    'Not feeling well'
17  )
18  rand=$[ $RANDOM % ${#EXCUSES[@]} ]
```

Le shell

- **Le shell est un programme permettant d'interagir avec les services fournis par un système d'exploitation**



  - **Dans ce cours, nous étudions le `shell` en mode texte `Bash`**
    - En mode texte car permet d'écrire des scripts !
    - **`Bash`** n'est qu'un shell parmi de nombreux autres shells (`bash, tcsh, zsh, ksh, cmd.exe…`)

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Le shell

- **Pourquoi apprendre le langage du shell**
  - Le shell est l'interface de tous les jours en UNIX. Bien connaître son shell permet d'économiser beaucoup d'efforts
  - Le shell est universel: peu importe le système UNIX, vous pouvez être certain de retrouver **sh** et **csh**
  - C'est facile de programmer en shell
    - Par rapport par exemple à C; le shell n'a pas été conçu pour être minimal ou théoriquement élégant; il a été conçu pour être flexible et pratique

Le shell

- **`Bash` : un acronyme pour « Bourne-Again shell » et un jeu de mots sur le désormais classique Bourne shell de Stephen Bourne**
  - Le standard de facto pour la programmation de scripts sur la plupart des systèmes UNIX

Bash

- **Interpréteur de commandes**
  - Lit des commandes (à partir du terminal ou d'un fichier)
  - Exécute les commandes
  - Écrit les résultats sur son terminal d'attache
- **`Bash` définit un langage, appelé le langage `bash`**
  - Structures algorithmiques classiques (if, while, for, etc.)
  - Variables
- **Accès rapide aux mécanismes offert par le noyau du système d'exploitation (tube, fichiers, redirections, …)**
  - ifconfig
  - rm, mv, mkdir
  - systemctl
  - ...

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Script `bash`

- **Un script `bash` est un fichier de type texte contenant une suite de commandes shell, exécutable par l'interpréteur (ici le programme /bin/bash)**
  - **Un script peut être lancé en ligne de commande, comme dans un autre script**
  - **Modifiable par un éditeur de texte**
  - **Un programme `bash` doit être rendu exécutable avec**

```
chmod u+x mon_script.sh
```

  - **Par convention, les noms de script sont suffixés par l'extension « .sh »**
- **Invocation du script nommé `mon_script.sh` avec**

```
./mon_script.sh
```

```
./mon_script.sh arg1 arg2
```

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Structure d'un script `bash`

- **Première ligne :**

```
#!/bin/bash
```

  - **#! : indique au système que ce fichier est un ensemble de commandes à exécuter par l'interpréteur dont le chemin suit**
  - **/bin/bash : lance bash**

- **Puis séquence structurée de commandes shell**

```
#!/bin/bash

commande1
commande2
```

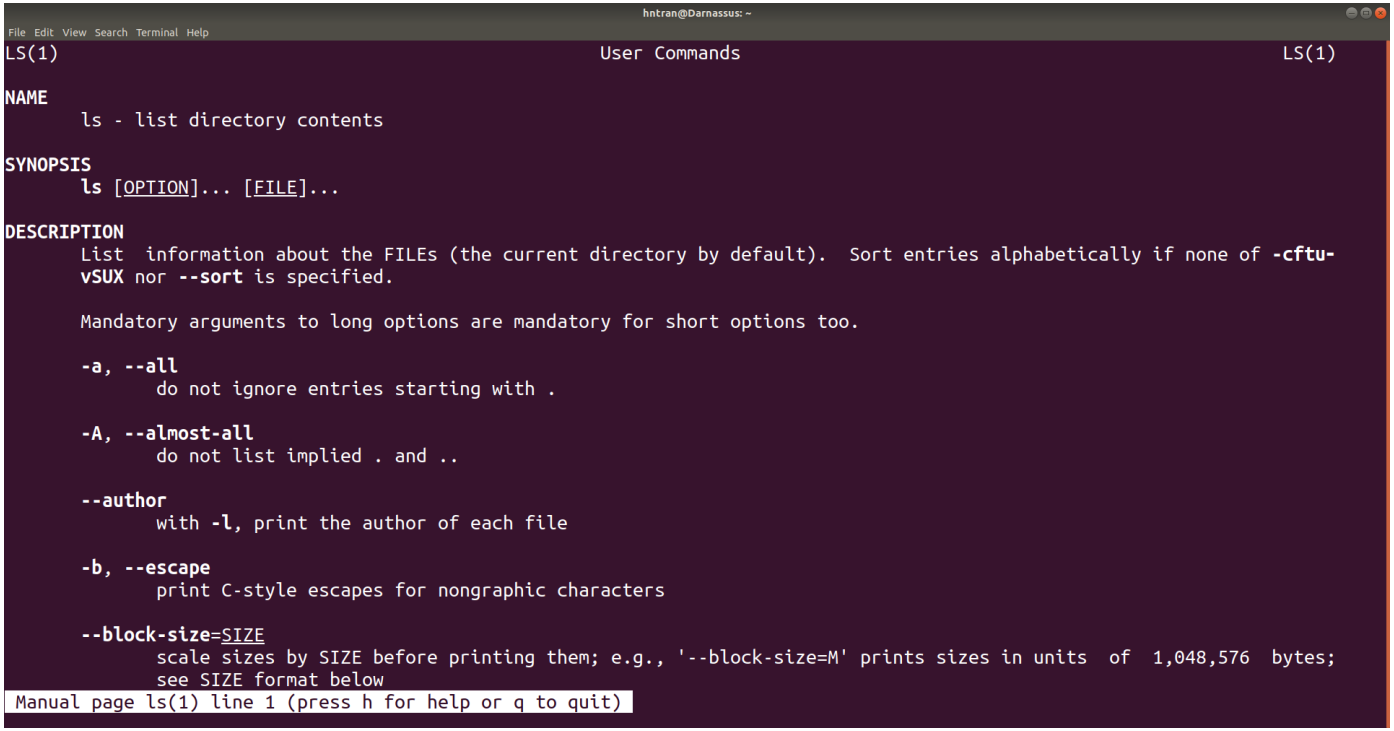
- **Sortie implicite du script à la fin du fichier**
  - **Sortie explicite avec la command `exit`**

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

La première commande à connaître

- `man cmd`
  - `man` pour manuel : donne de l'aide
  - `cmd` est la commande dont on veut consulter le manuel

```
$ man ls
```



Caractères spéciaux de bash

- **Caractères spéciaux**
  - **\ ' ` " > < $ # * ~ ? ; ( ) { }**
    - (' est appelé quote ou apostrophe alors que ` est appelé antiquote ou accent grave)
  - **Explication de chacun donnée dans la suite du cours**

- **Désactiver l'interprétation des caractères spéciaux**
  - **\ : désactive l'interprétation spéciale du caractère suivant**
  - **'…' : désactive l'interprétation dans toute la chaîne**
  - **"…" : seuls sont interprétés les caractères $ \ ` (accent grave)**

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Variables `bash`

- **Déclaration/affectation avec =**
  - **`ma_var=valeur`**
  - **Pas d'espace blanc dans la chaine "`ma_var=valeur`"**
- **Consultation en préfixant du caractère $**
  - **`$ma_var`**
  - **Pas d'espace blanc dans "`$ma_var`"**
- ***Saisie interactive***
  - **`read var1 var2 ... varn a b c ...`**
    - Lecture d'une ligne saisie par l'utilisateur (jusqu'au retour chariot)
    - Le premier mot va dans var1
    - Le second dans var2
    - Tous les mots restants vont dans varn

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Variables bash

- Définir des variables dans un fichier (configuration)

```
MACHINE_LISTE=hosts
IF=enp0s3
DOMAINE_IP="ubo.local"
```

- Inclure les variables dans un autre fichier

```
#!/bin/bash

. param
```

Variables `bash` - exemple

```
$ a=42
$ echo $a
42
$ s='Bonjour, monde!!!'
$ echo $s
Bonjour, monde!!!
$ read x Ceci est une phrase
$ echo $x
Ceci est une phrase
$ read x y Ceci est une phrase
$ echo $x
Ceci
$ echo $y
est une phrase
```

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Fonctions `bash`

- **Pour déclarer une fonction, on utilise la syntaxe suivante**

```
maFonction()
{
   instructions
}
```

- **Pour appeler une fonction, on utilise la syntaxe suivante**

```
maFonction param_1 param_2 ... param_n
```

- **Paramètres passés à la fonction**
  - **À l'intérieur de la fonction, les paramètres sont représentés, respectivement, par les variables $1, $2,... , $n**

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Fonctions bash

```
#!/bin/bash

# déclaration d'une fonction
maFonction()
{ varlocal="je suis une fonction"
  echo "$varlocal"
  echo "Nombres de paramètres : $#"
  echo $1
  echo $2
}

# appel de ma fonction
maFonction "Hello" "World!"
```

```
je suis la fonction
Nombres de paramètres : 2
Hello
World!
```

Script bash - exemple

```
#!/bin/bash
. param

ma_machine=$1
echo $ma_machine
ip=`grep $ma_machine $MACHINE_LISTE | cut -d ' ' -f 1`
echo $ip
ifconfig $IF $ip
```

```
#!/bin/bash

systemctl enable nfs-server
systemctl start nfs-server
systemctl stop firewalld.service
```

Schéma algorithmique séquentiel

- **Suite de commandes les unes après les autres**
  - **Sur des lignes séparées**
  - **Sur une même ligne en utilisant le caractère point-virgule (;) pour séparateur**

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Schéma alternatif (`if`)

- **Schéma alternatif simple**
  - **Si alors … sinon (si alors … sinon …)**
  - **`elif` et `else` sont optionnels**

```
if cond; then
    cmds
elif cond; then
    cmds
else
   cmds
fi
```

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Conditions de test

- **Tests sur des valeurs numériques**
  - `[ n1 -eq n2 ]` : vrai si n1 est égal à n2
  - `[ n1 -ne n2 ]` : vrai si n1 est différent de n2
  - `[ n1 -gt n2 ]` : vrai si n1 supérieur strictement à n2
  - `[ n1 -ge n2 ]` : vrai si n1 supérieur ou égal à n2
  - `[ n1 -lt n2 ]` : vrai si n1 inférieur strictement à n2
  - `[ n1 -le n2 ]` : vrai si n1 est inférieur ou égal à n2
- **Tests sur des chaînes de caractères**
  - `[ mot1 = mot2 ]` : vrai si mot1 est égale à mot2
  - `[ mot1 != mot2 ]` : vrai si mot1 n'est pas égale à mot2
  - `[ -z mot ]` : vrai si mot est le mot vide
  - `[ -n mot ]` : vrai si mot n'est pas le mot vide
- **Les espaces blancs sont essentiels !**

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Schémas itératifs

- **Boucles**
  - **while**
    - Tant que … faire …
    - Mot clé break pour sortir de la boucle

```
while cond; do
    cmds
done
```

```
x=10
while [ $x -ge 0 ]; do
    read x
    echo $x
done
```

  - **for**
    - Pour chaque … dans … faire …
    - var correspond à la variable d'itération
    - liste : ensemble sur lequel var itère

```
for var in liste; do
    cmds
done
```

```
for var in 1 2 3 4; do
    echo $var
done
```

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Arguments d'une commande

- `mon_script.sh arg1 arg2 arg3 arg4 …`
  - **chaque mot est stocké dans une variable numérotée**

mon_script.sh	arg1	arg2	arg3	arg4	…
"$0"	"$1"	"$2"	"$3"	"$4"	…

  - "$0" : toujours le nom de la commande
  - • "$1" … "$9" : les paramètres de la commande
  - • $# : nombre de paramètres de la commande
  - • "$@" : liste des paramètres : "arg1" "arg2" "arg3" "arg4" …

```
#! /bin/bash
for i in "$@"; do
 echo $i
done
```

```
$./mon_echo.sh
$./mon_echo.sh toto titi
toto
titi
$
```

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Imbrication de commandes

- **Pour récupérer le texte écrit sur le terminal par une commande dans une chaîne de caractères**
  - `$(cmd)`
  - **Attention à ne pas confondre avec `$cmd` qui permet l'accès à la valeur de la variable `cmd`**

```
$ date
lundi 27 juillet 2015, 12:47:06 (UTC+0200)
$ echo "Nous sommes le $(date). "
Nous sommes le lundi 27 juillet 2015, 12:47:06
(UTC+0200).
$
```

Reference : CSC3102 - Introduction aux systèmes d'exploitation. Élisabeth Brunet et Gaël Thomas

Projet préparation

- **TP1**
  - **man**
  - **cp, rm, mv, mkdir**
  - **echo, cat, grep, cut, sed**
  - **if then else**
  - **boucle for et while**

Devoir 1
Expliquer et donner un exemple d'utilisation pour chaque commande dans la liste
Date limite : 27/01 23:59
Zone de dépôt : Moodle

Plan

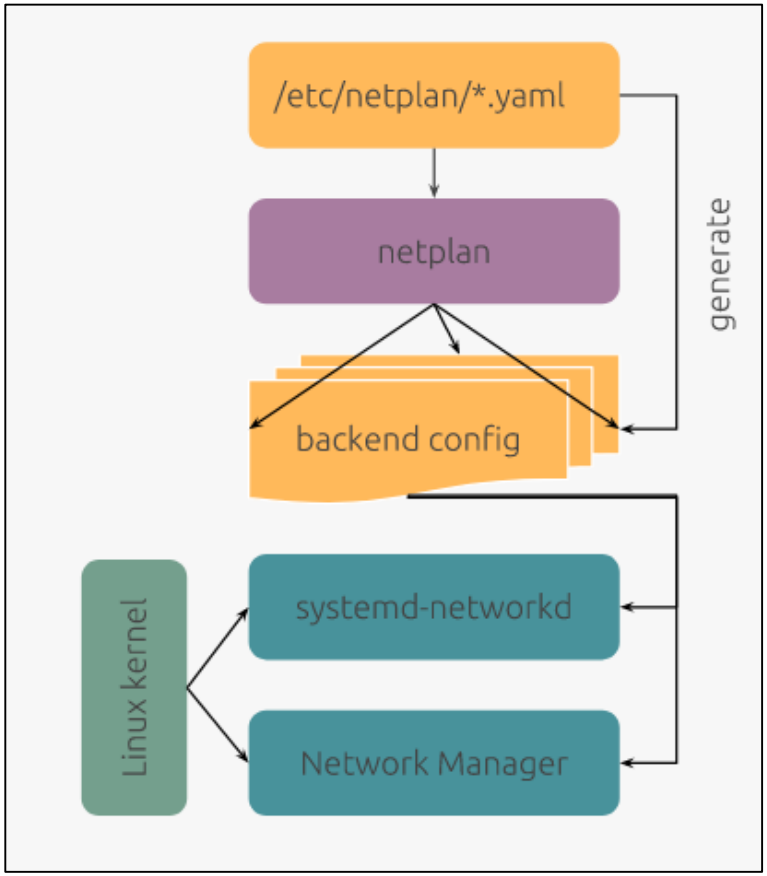
- Introduction
- Configuration - Introduction aux scripts BASH
- **Configuration - Réseau**
- **Accès distant aux ressources de stockage**
- **Infrastructure d’annuaires**
  - **Notion d'annuaires informatiques**
  - **Network Information System (NIS)**
  - **Domain Name Server (DNS)**
  - **OpenLDAP, Active Directory**
  - **Sélection du service d'annuaires**
- **Outils de travail collaboratif**

Configuration réseau

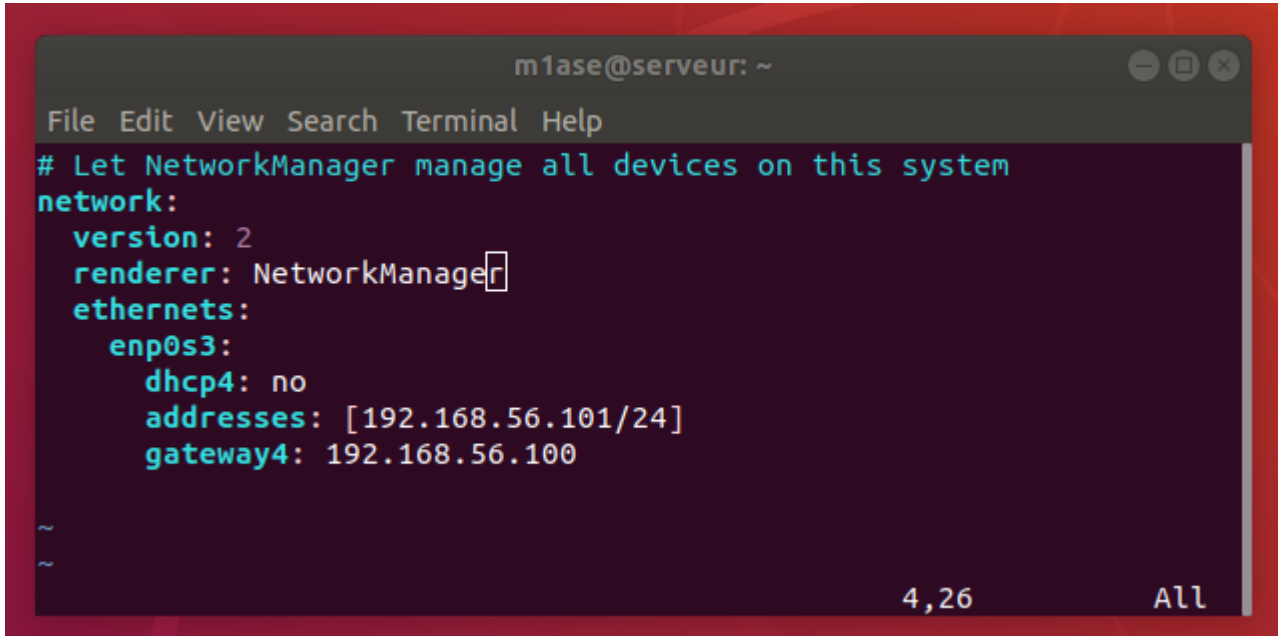
- **Definition**
  - **"*La configuration réseau est l'ensemble des caractéristiques d'un réseau donné. Autant les caractéristiques physiques telles que la connectique que les caractéristiques logiques telles que les protocoles utilisés, les adresses IP, ainsi que le nom de chaque machine branchée au réseau.*"**

Configuration réseau : un exemple

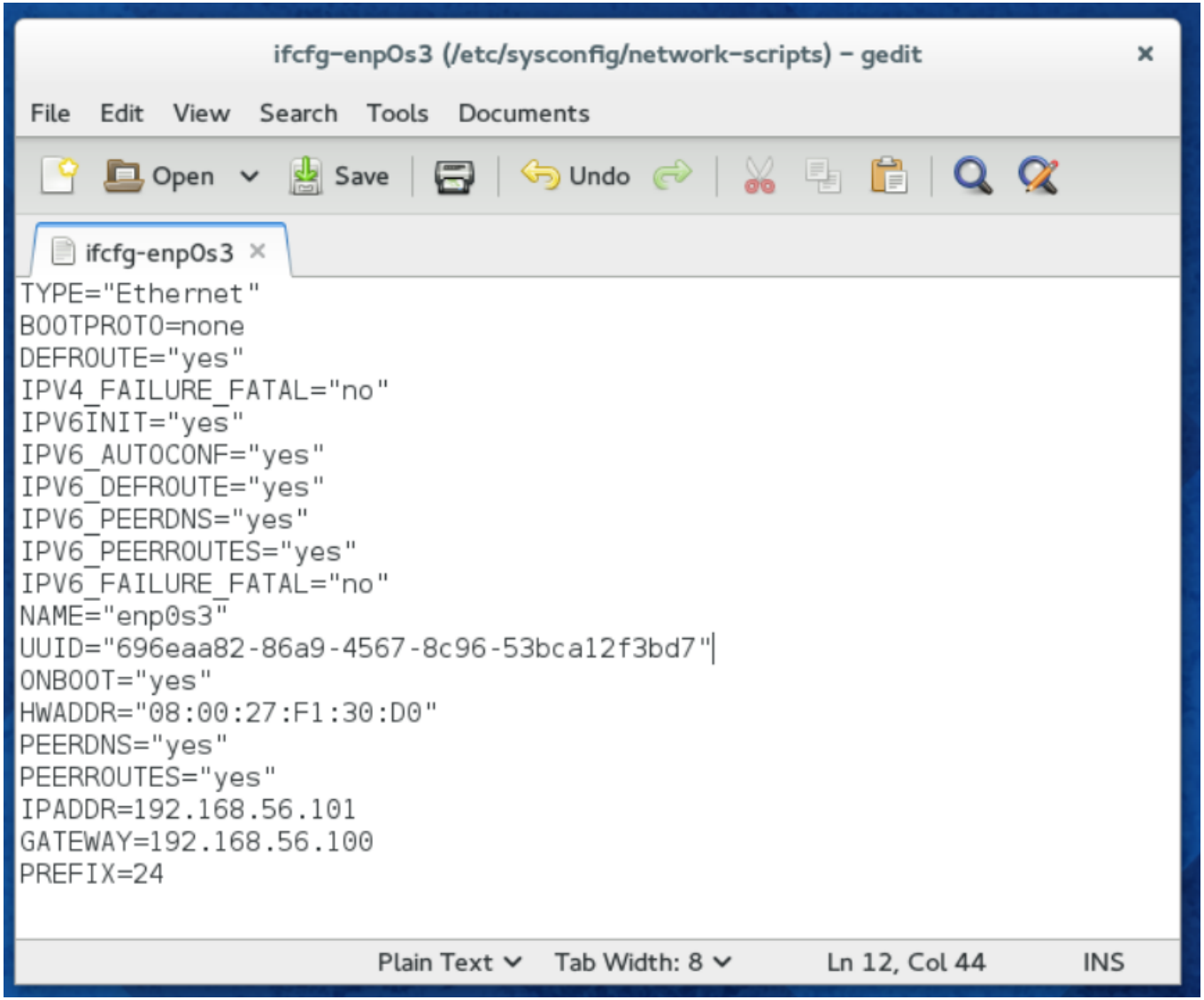
- **Scripts de configuration de l'interface réseau**
- **La version > 18.04 d'Ubuntu utilise Netplan pour la configuration réseau**
  - **https://netplan.io/**
  - **La configuration se trouve dans le dossier /etc/netplan**
- **< 18.04**
  - **Network scripts**
  - **/etc/sysconfig/network-scripts**



Configuration réseau : un exemple



Configuration réseau : un exemple



Configuration réseau : un exemple

- **Configuration de netplan**
  - **Plusieurs exemple disponible https://netplan.io/examples/**

```
network:
    version: 2
    renderer: networkd
    ethernets:
        enp3s0:
            addresses:
                - 10.10.10.2/24
            nameservers:
                search: [mydomain, otherdomain]
                addresses: [10.10.10.1, 1.1.1.1]
            routes:
                - to: default
                  via: 10.10.10.1
```

les espaces blancs sont importants !

  - **Pour plus d'information : https://netplan.io/reference**

Configuration réseau : un exemple

- network : Le "top-level node" dans une configuration netplan
- version : La version du YAML
- renderer : networkd ou NetworkManager
- ethernets : type d'appareil
- enp0s3 : nom de l'interface réseau
- addresses : adresse IP statique de l'interface
- nameservers : adresse du de serveur DNS

Configuration réseau : hostname

- **Hostname**
  - **/etc/hostname**
  - **Hostname est également le programme utilisé soit pour définir le nom d'hôte, soit pour visualiser le nom actuel d'hôte ou de domaine du système.**
    - Ce nom est utilisé par de nombreux programmes de réseaux pour identifier la machine. Le nom de domaine est également utilisé par NIS/YP.

Configuration réseau : name lookup

- **Name Lookup**
  - **Solution 1 : Un serveur DNS dédié**
    - Plusieurs solutions existent : BIND, Dnsmasq, Windows Server, …
    - Probablement pas assez de temps pour installer et tester dans le cadre du projet ☹
  - **Solution 2 : Serveur DNS locale**
    - Fichier /etc/hosts

Configuration réseau : fichier /etc/hosts

- **Un fichier de texte simple qui associe les adresses IP avec les noms d'hôtes, une ligne par adresse IP**

```
Adresse_IP Nom_officiel [Alias...]
```

  - **Dans les systèmes modernes, même si la table des hôtes a été remplacée par DNS, ce mécanisme est encore largement employé pour :**
    - **Initialiser une machine** : Beaucoup de systèmes ont un petit fichier contenant le nom et l'adresse des hôtes important sur le réseau local. Ceci est utile lorsque le DNS n'est pas en marche, notamment lors de la mise en route des systèmes.
    - **NIS** : Les sites employant NIS utilisent la table d'hôtes comme entrée pour la base de données des hôtes NIS.
    - **Noeud isolés** : Les petits sites, isolés des réseaux importants emploient la table d'hôtes à la place du DNS. Si les informations locales changent rarement, et si le réseau n'est pas connecté à Internet, le DNS n'offre pas beaucoup d'intérêt

Configuration réseau : fichier /etc/hosts

- **Un fichier de texte simple qui associe les adresses IP avec les noms d'hôtes, une ligne par adresse IP**

```
Adresse_IP Nom_officiel [Alias...]
```

```
127.0.0.1         localhost
192.168.56.101    serveur.ubo.local   serveur
192.168.56.102    client.ubo.local    client
192.168.56.103    esclave.ubo.local   esclave
```