Technologies

de l'internet

FRANZ GIRARDIN

Université de Montréal

CHARGÉE DE COURS

Philippe Langlais

Département d'Informatique et de Recherche Opérationnelle

Notes de cours

TPC/IP, sciptage

Montréal 2025

3 août 2025

Table des matières

1 Modèle TCP-IP

Définition 1.1 (TCP-IP). Architecture simplifiée en **4 couches** décrivant comment les données sont échangées sur un réseau, notamment l'Internet, en référençant les technologies nécessaires.

$$\underbrace{\text{Application}}_{\texttt{HTTP, FTP, SMTP, SSH, Telnet}} + \underbrace{\text{Transport}}_{\texttt{TCP, UDP}} + \underbrace{\text{Internet}}_{\texttt{IP, ICMP}} + \underbrace{\text{Réseau}}_{\texttt{Ethernet, Wi-Fi}}$$

La couche *application* gère les interactions utilisateurs ; *transport* permet la communication machine ; *internet* sert à l'adressage et routage des paquets ; et la couche *réseau* est responsable du transport physique des données.

1.1 SSH

SSH est un protocole de réseau qui permet d'établir une connexion sur une machine distante afin d'exécuter des opérations en ligne de commande et d'effectuer le transfert de fichiers via des technologies auxiliaires telles SCP et SFTP.

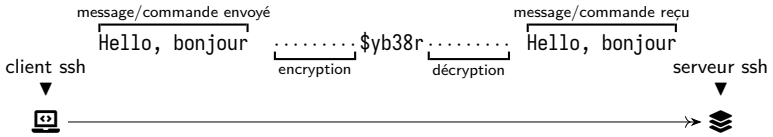


Figure 1.1 – Transmission sécurisée d'un message via SSH

1.2 FTP, SFTP, SMTP, POP, IMAP, UDP

SFTP (*Secure File Transfer Protocol*) date de 1995 et est un successeur de FTP (1970). Contrairement à son prédecesseur, SFTP permet le transport sécuritaire des données de fichiers en communicant sur le port 22 et en utilisant un seul canal chiffré et authentifié par SSH.

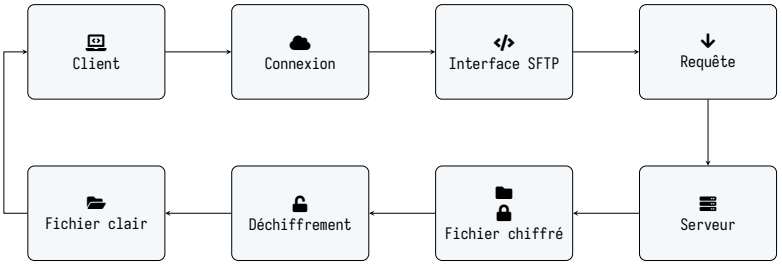


Figure 1.2 – Étapes d'un transfert de fichier sécurisé via SFTP

SMTP (*Simple Mail Transfer Protocol*) est un protocole standard de l'Internet destiné à l'*envoi* et au relais de courriels électroniques. Il fonctionne sur le port 25, en mode non sécurisé, sauf lorsqu'il est encapsulé via des alternatives chiffrées telles que SMTPS ou TLS.

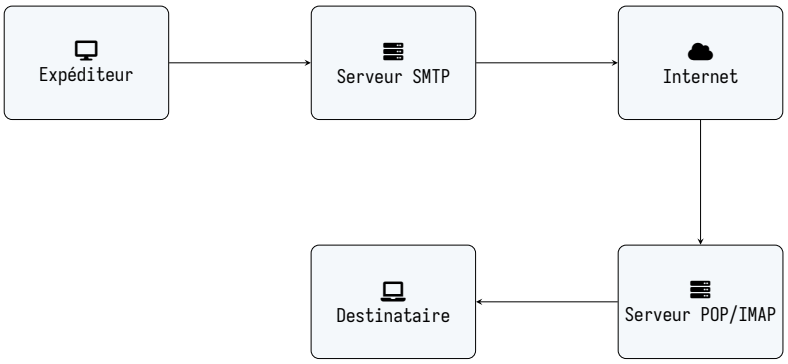


Figure 1.3 – Chemin de transmission d'un courriel via SMTP et POP/IMAP

Notons que SMTP est un protocole à *flux unidirectionnel* : il est *utilisé exclusivement pour l'envoi de messages*. Il ne permet ni la réception, ni la consultation du courrier par le destinataire.

POP (*Post Office Protocol*) est un protocole de la couche application du modèle qui est utilisé par les clients de messagerie pour récupérer ou pour supprimer les courriels depuis un serveur mail. Le flux est unidirectionnel. Il permet uniquement de stocker localement le courriel disponible sur un serveur. Le protocole opère de façon non sécurisée sur le port 110 ou de façon sécurisée sur le port 995.

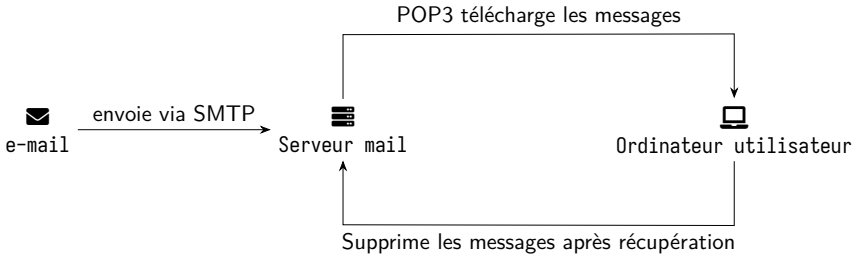


Figure 1.4 – Fonctionnement du protocole POP3 pour la réception des courriels

IMAP (*Internet Message Access Protocol*) est un protocole de la couche *application* du modèle qui permet lire, organiser les courriels à distance sur le serveur. Le protocole a un *flux bidirectionnel* : les changements effectué localement sont répercuté sur le serveur. IMAP fonctionne via le port 143 par défaut (non sécurisé) ou de façon sécurisée via le port 993 en TLS.

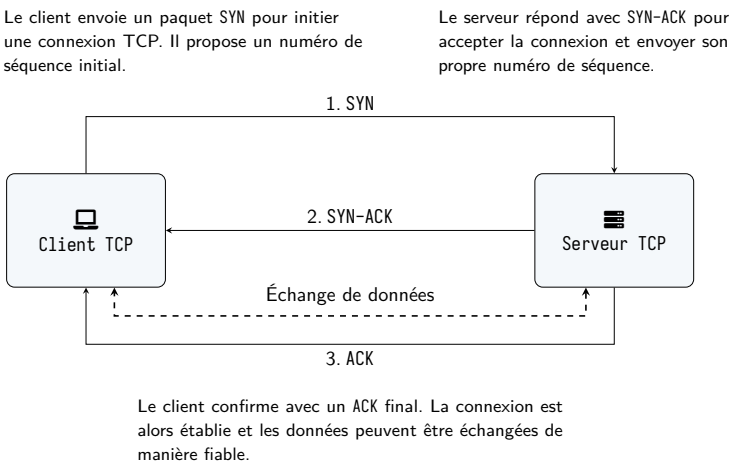


Figure 1.6 – Établissement d'une connexion TCP selon le *three-way handshake*

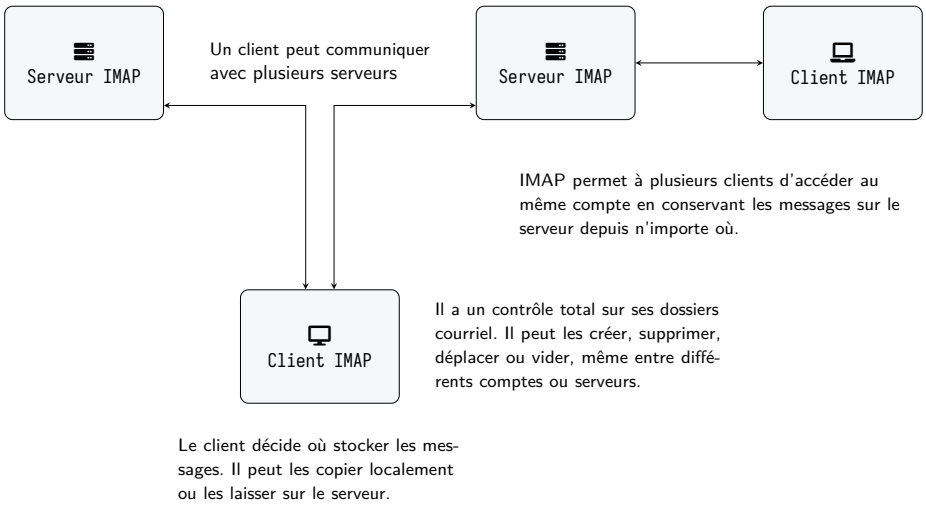


Figure 1.5 – Architecture IMAP avec accès simultané à plusieurs serveurs

TCP *Transmission Control Protocol* est un protocole de la couche *transport* qui foutnit un service de transimission fiablle, ordonné et avec un contrôle de flux entre hôtes sur un réseau IP.

Remarque 1.2 (Modèle TCP/IP et protocole TCP)**.** Il faut savoir distinguer le modèle TCP/IP qui fait référence à un concept décrivant les quatres couches de technologies impliquées dans le système TCP/IP de la *technologie* TCP qui est un protocole à part entière faisant partie de la couche transport du modèle TCP/IP. ⌟

UDP (*User Datagram Protocol*) est un protocole de la couche *transport* de la suite IP, conçu pour envoyer des datagrammes—des unités de données échangées sur les réseau à commutation de paquets—sans établir de connexion préalable et sans garantie de livraison.

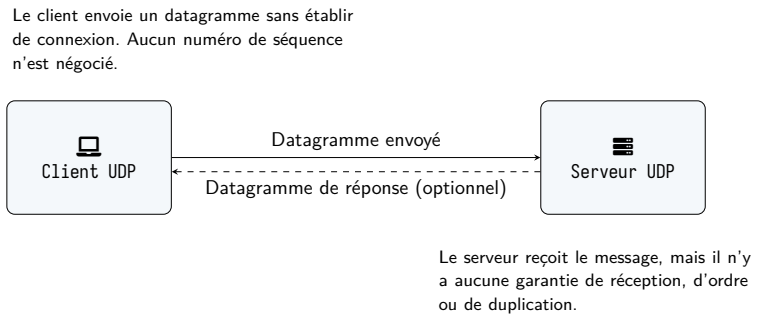


Figure 1.7 – Envoi de datagrammes sans connexion avec UDP

UDP est un protocole léger, sans connexion, adapté aux applications qui privilégient la rapidité à la fiabilité, comme le streaming ou le DNS.

1.3 Datagramme

Un datagramme est une unité de données qui comporte deux compsant structurels, soit une en-tête aussi appelée *header* et une charge utile aussi appelé *payload*.

L'en-tête contient les informations de routate (adresse, longueur, type, etc.), alors que la charge utile transporte les données de l'utilisateur. Dans le contexte de la suite TCP/IP, les paquets IP sont essentiellement des datagrammes IP.

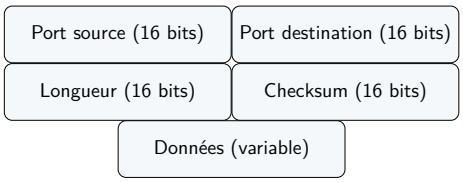


Figure 1.8 – Format de l'en-tête UDP

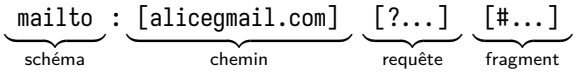
1.4 Procesus sans connexion UDP et TCP

Contraitement à TCP qui effectue un *handshake* en trois temps pour négocier l'ouverture d'une session avant d'envoyer des données, UDP se contente d'envoyer des datagrammes de façon aveugle. Par conséquent, l'expéditeur n'attend pas de confirmation (SYN) que le destinataire est prêt à recevoir.

Par ailleurs, UDP ne prévoit ni accusé de réception (ACK), ni réordonnancement, ni de retransmission automatique. Ainsi, il n'y a pas de garantie que les données atteignent leur destination ou que leur intégrité soit préservée. UDP est donc considéré comme un protocole *connectionless* et *unreliable*—sans garantie de fidelité.

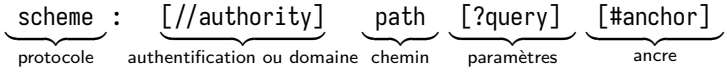
1.5 Identificateur et localisateur

Une URI ou *Universal Ressource Identifier* est un identifiant générique de ressource sur le Web. La syntaxe générale d'un URI est la suivante :



Il existe **plusieurs type d'URI** : les **URN** pour les technologies de courrier, les URN pour les numéro d'ISBN, et les **URL** pour les pages Web sont des exemples d'URI.

Concept 1.3 (URL). Une URL est composée, entre autres, d'un *schéma*, p. ex. HTTP, d'un *nom de domaine*, d'un *chemin*, de *paramètres* et d'une *ancre*.



C'est un **type particulier** d'URI qui, en plus d'identifier, localise la ressource et indique comment y accéder.

1.6 Fonctionnement du Web et URI

Web = URI + HTTP + HTML

Le modèle ci-haut permet de mettre en évidences les trois briques fondamentales du Web. Le protocole **HTTP** régit la communication entre différentes sources et destinataires selon le modèle *client-serveur*. Le langage **HTML** permet la conception de page Web, et les technologies d'**URI** permettent d'associer une ressource telle qu'une page à des caractéristiques d'identification, de localisation et destinations.

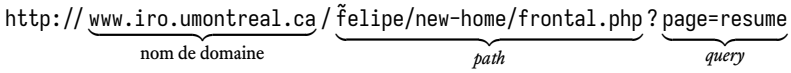


Figure 1.9 – Exemple d'URI

1.7 DNS

Le DNS est un système distribué utilisé pour *traduire les noms de domaine* lisibles par l'humain en adresses IP compréhensibles par les machines.

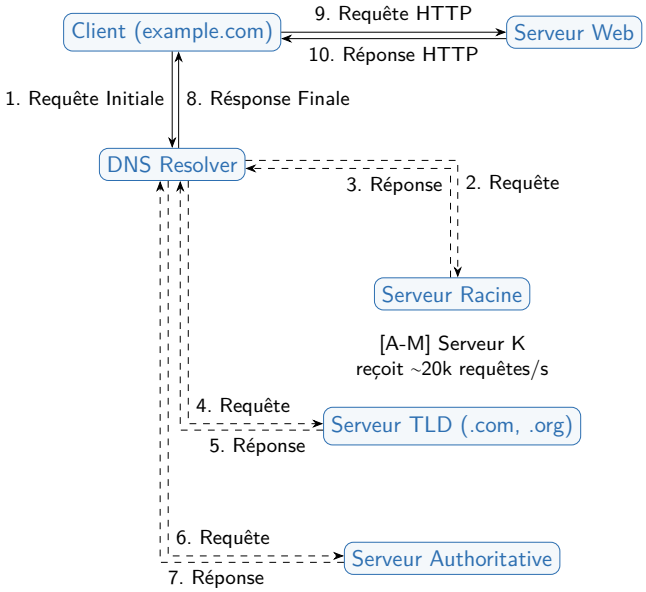


Figure 1.10 – Étapes de requête d'une résolution DNS

La résolution DNS débute par la requête initiale du client vers son *resolver* local configuré par le FAI (étape 1). Le resolver envoie une requête itérative au serveur racine (étape 2), qui répond en fournissant l'adresse d'un TLD adapté (étape 3). Le resolver interroge alors le serveur TLD responsable des adresses *.com* et *.org*, par exemple (étape 4), lequel renvoie l'adresse du serveur *authoritative* pour le domaine (étape 5). Celui-ci retourne l'enregistrement A/AAAA demandé (étape 7). Le resolver assemble les réponses puis envoie la réponse finale au client (étape 8). Le client initie ensuite une connexion HTTP vers le serveur Web obtenu (étape 9) et reçoit la réponse HTTP (étape 10).

1.8 DNS Resolver

Le *stub resolver* est intégré à l'OS et transmet les requêtes DNS aux resolvers récursifs (BIND, Unbound, services FAI ou tiers publics). Le resolver est avant tout un logiciel, côté client et côté serveur. Certains routeurs ou appliances réseau intègrent un resolver optimisé en firmware.

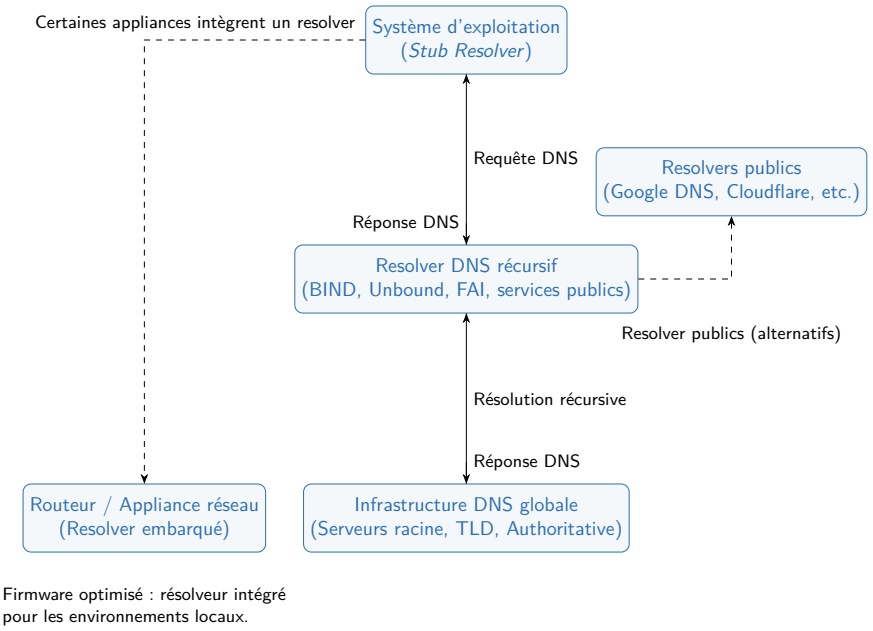


Figure 1.11 – Fonctionnement du DNS Resolver (Stub, récursif, public, intégré)

1.9 Emplacement du Resolver

Le stub resolver vit dans chaque poste utilisateur. Le resolver récursif peut résider dans le routeur local, chez le FAI ou chez un fournisseur DNS public (1.1.1.1).

1.10 Serveur Racine

Le serveur racine gère la zone « . » et renvoie les adresses anycast des serveurs TLD responsable des adresses .com, .org, .fr, etc., sans fournir directement l'enregistrement final.

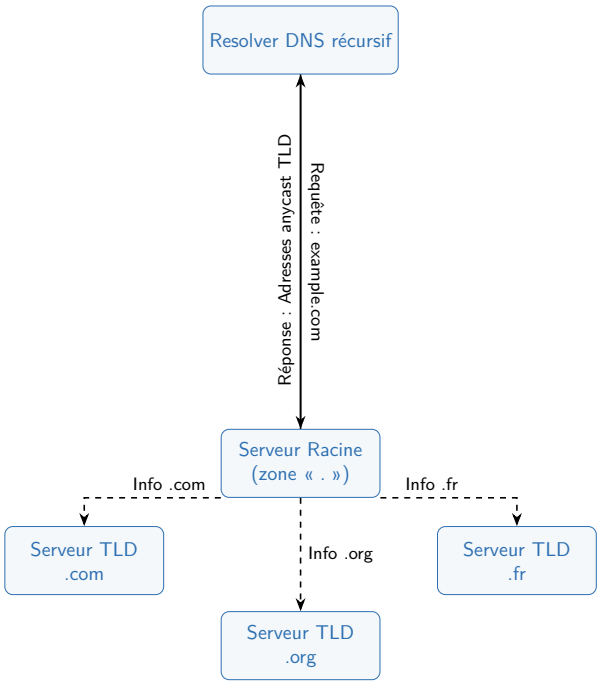


Figure 1.12 – Fonctionnement du Serveur Racine DNS

1.11 Enregistrements A et AAAA

Les enregistrements A associent un nom de domaine à une adresse IPv4, tandis que AAAA fournit une adresse IPv6. Lorsqu'un resolver interroge un serveur DNS pour obtenir les enregistrements A/AAAA, il reçoit l'adresse IP cible pour établir la connexion.

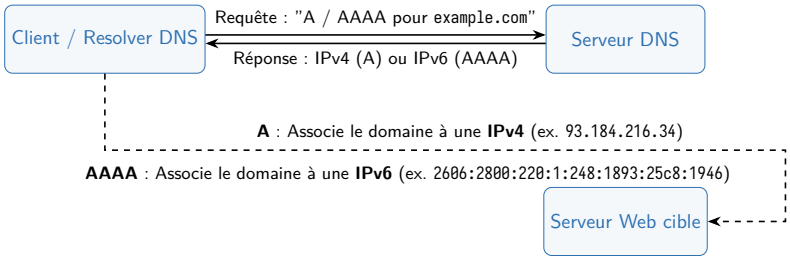


Figure 1.13 – Résolution DNS avec enregistrements A (IPv4) et AAAA (IPv6)

1.12 Serveur TLD

Un serveur TLD gère un *Top-Level Domain* (.com, .org, .fr). Il délègue la résolution des noms aux serveurs *authoritative* via des enregistrements NS.

1.13 Serveur authoritative

Le serveur *authoritative* détient la copie officielle des enregistrements DNS d'une zone (maître ou secondaire). Il répond de manière définitive aux requêtes sans délégation supplémentaire.

1.14 Avantages de la répartition

Cette hiérarchie garantit la scalabilité en partageant la charge, améliore la performance grâce aux caches et à la répartition géographique, renforce la robustesse via la redondance et permet une gestion déléguée et sécurisée.

1.15 HTTP

Protocole basé sur le modèle client-serveur où le C envoie une requête au S et le S répond avec les ressources demandées. Plusieurs proxy intermédiaires peuvent intervenir entre l'envoie et la réception.

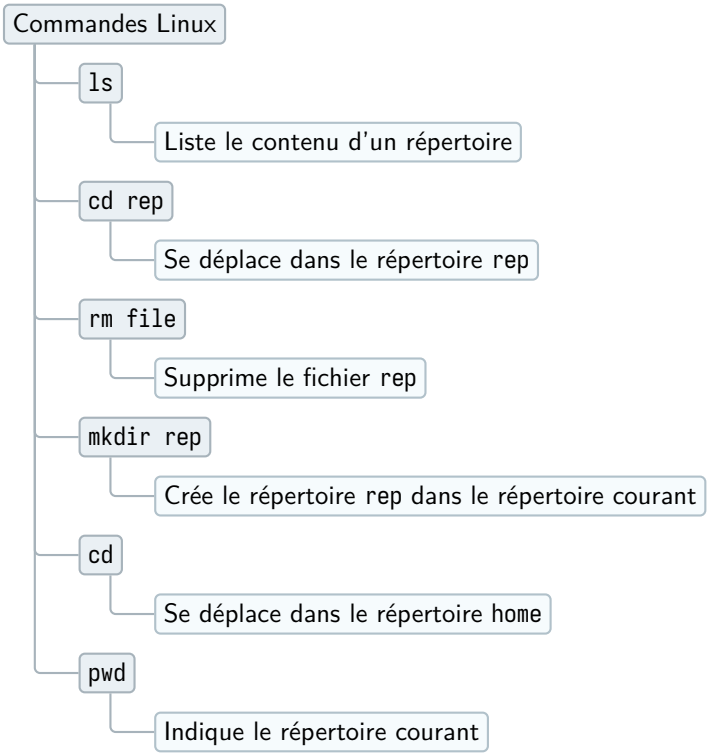
Sans état : HTTP est un des rares protocoles sans état . SMTP , IMAP et FTP maintiennent une session durant laquelle des informations de contexte sont échangées.

Types de requêtes : GET récupère la ressource ; POST crée une ressource en envoyant des données ; PUT remplace ou crée la ressource ; DELETE la supprime ; HEAD renvoie uniquement les entêtes.

2 Scriptage

La connaissance de commandes UNIX et une compréhension générale du langage bash évite souvent d'avoir à programmer des scripts élaborés qui peuvent être remplacé par des commandes simples mais sophistiquées.

2.1 Commande linux de navigation et de gestion de fichiers



2.1.1 Pipeline de redirection

Concept 2.1 (Pipeline). La pipeline dénotée par la syntaxe `|` permet de diriger la sortie d'une première commande sur l'entrée standard d'une seconde commande afin que l'*output* de la première serve d'*input* à la seconde.

Exemple 2.2 (Utilisation d'une pipeline pour inverser l'affichage d'un string). La commande `rev` renverse le texte en *input* ligne à ligne.

```
echo "bonjour" | rev
```

On contaste que, contrairement à `tac`, la commande `rev` ne modifie par l'ordre des lignes. Par ailleurs, `rev` n'inverse pas tous le contenu d'un fichier, mais plutôt **chaque ligne individuellement**. `rev`.

Note 2.3 (Directionnalité). L'inversion de l'ordre de la commande n'est pas possible puisque `rev` lit uniquement sur des entrées standard (`input` d'une pipeline) ou un fichier, et ne prend pas directement d'argument de texte en ligne de commande :

```
rev "bonjour" | echo
```

L'exemple ci-haut **ne respecte pas la syntaxe**. D'ailleurs, `echo` s'attend à un argument qui le succède.

Concept 2.4 (Redirection en `csh`). Le symbole de redirection `>` redirige la sortie standard vers un fichier. Si le fichier existe déjà, **il est écrasé**.

L'opérateur `>!` Force la redirection de la sortie standard vers un fichier. Cela écrase le fichier existant **sans avertissement**.

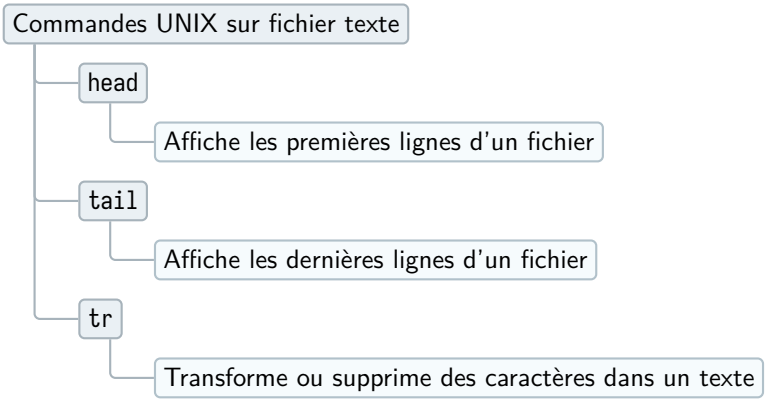
L'opérateur `>>` ajoute la sortie standard à la fin d'un fichier existant. Si le fichier n'existe pas, il est créé.

Exemple 2.5. Redirection vers un fichier

```
echo "bonjour" | rev > fichier.txt
```

La commande `>` permet d'enregistrer un *input* dans le fichier `fichier.txt`.

2.2 Manipulation de fichiers textes



2.3 La commande tr

Définition 2.6 (Commande tr). La commande `tr` est utilisée pour transformer ou supprimer des caractères dans une entrée standard.

2.3.1 Utilisation de tr et -s

Exemple 2.7. Découpe en mots une ligne avec ponctuation

```
head -n 1 zola1.txt | tr    '[[:punct:]]' ' '
                    | tr    '[[:space:]]' '\n'
                    | tr -s '[[:space:]]'
```

Cette commande découpe la première ligne de `zola1.txt` en mots en remplaçant les ponctuations par des espaces, divisant chaque mot sur une nouvelle ligne et éliminant les espaces redondants.

L'utilisation de `tr` est justifiée par le fait que la commande différencie correctement un mot suivi d'une virgule ou d'un point. Par exemple, on obtient la transformation suivante :

`Bourse,` ⟶ `Bourse`

Note 2.8. L'option `-s` supprime les **répétitions successives** des caractères spécifiés dans l'ensemble donné. Ainsi :

```
echo "aaa   bbb    ccc" | tr -s ' '
```

engendre le *output* `"aaa bbb ccc"` en supprimant les répétitons d'espaces, soit `' '`.

Exemple 2.9. Découpage sélectif des mots alphabétiques

```
head -n 1 zola1.txt | tr -sc '[[:alpha:]]' '\n'
```

Cette commande découpe la première ligne de `zola1.txt` en mots, remplaçant *tous caractères non alphabétiques*, y compris ponctuation, par des sauts de ligne.

Cette approche est plus concise et efficace. Elle ne nécessite pas de multiples appels à `tr` et évite les mots collés à une ponctuation.

2.3.2 Utilisation de tr et -sc

Exemple 2.10 (Combinaison avec l'option complément -c). Dans l'exemple ci-bas, les options `-sc` combinent deux fonctionnalités. L'option `-c` prend le complément de l'ensemble spécifié qui est `:alpha:`, soit **tout sauf les lettres alphabétiques**, et l'option `-c` compresse les répétitions des caractères du complément.

```
echo "abc123@@@def456" | tr -sc '[:alpha:]' '\n'
```

Ainsi, on obtient la transformation suivante :

```
1.abc123@@@def456  ──────────────────→  1.abc
                                        2.def
```

Ainsi, les caractères non alphabétiques sont remplacés par des sauts de ligne compressés.

2.4 La commande grep

Concept 2.11 (Commande grep). La commande grep est utilisée pour recherches des lignes correspondant à un *motif* dans un fichier ou dans une entrée standard.

Exemple 2.12 (Recherche insensible à la casse avec `grep -i`). Cette commande recherche toutes les lignes contenant *bonjour* dans le fichier spécifié, sans tenir compte de la casse.

```
grep -i "bonjour" fichier.txt
```

Exemple 2.13. Recherche du motif ven insensible à la casse

```
head -n 60 zola1.txt | tr '[[:space:]]' '\n' | grep -i ven
```

Cette commande extrait les 60 premières lignes du fichier `zola1.txt`, divise chaque mot sur une nouvelle ligne en remplaçant les espaces par des sauts de ligne, puis recherche toutes les occurrences du motif ven sans tenir compte de la casse.

Exemple 2.14. Recherche avec surlignage du motif ven

```
head -n 60 zola1.txt | tr '[[:space:]]' '\n' | grep --colour -i ven
```

Cette commande effectue la même recherche que l'exemple précédent, mais utilise l'option `--colour` pour surligner en couleur toutes les occurrences du motif ven, rendant les résultats plus visibles.

2.5 La commande awk

Définition 2.15 (Commande awk). La commande awk est un langage de traitement de texte utilisé pour analyser et manipuler des fichiers ou des flux de données structurés, ligne par ligne, en fonction de motifs et d'actions spécifiés.

Exemple 2.16. Filtrage de mots fréquents selon leur ordre

```
head -n 200 zola1.txt | tr '[[:space:]]' '\n'
                      | grep -i '^ven'
                      | sort | uniq -c
                      | sort -k1,1nr | awk '{print $2}'
```

Notons ici que `n` force le trie numérique plutôt que le tri par défaut en ASCII, afin que, par exemple, la chaîne «10»soit considéré plus grande que la chaîne «2». Par ailleurs, le mot clé `r` permet d'effectuer le tri en ordre inverse. L'option `-k1,1` quant à elle définit une «clé»de tri qui va du champ 1 au champ 1, c'est-à-dire que seule la première colonne est inspectée pour décider de l'ordre. Finalement, `$2` désigne le deuxième champ de chaque ligne. Ainsi, la commande `awk 'print $2'` permet d'afficher la deuxième colonne de chaque ligne.

Exemple 2.17. Filtrage des mots selon leur occurence

```
cat zola1.txt | tr '[[:punct:]]' ' '
              | tr '[[:space:]]' '\n'
              | grep -i '^ven'
              | sort | uniq -c
              | sort -k1,1nr
              | awk '$1 > 3 {print $0}'
```

Le mot clé `awk '$1 > 3 print $0'` affiche uniquement les mots avec plus de 3 occurences, avec leur fréquence.

2.6 La commande awk

Définition 2.18 (La commande wc). La commande wc ou *word count* est utilisé pour compter les lignes, le smots ou les caractères dans uen entrée standard ou un fichier.

Exemple 2.19. Compter les mots uniques

```
cat zola1.txt | tr      '[[:punct:]]' ' '
              | tr   -s '[[:space:]]'
              | tr      '[[:space:]]' '\n'
              | sort
              | uniq -c
              | wc   -l
```

La commande `uniq -c` regroupe les lignes identiques adjacentes. À titre d'exemple,

cette même commande engendrerait la transformation suivante

La commande `wc -l` quant à elle compte uniquement le nombre de lignes. Ainsi, dans l'exemple c