Franz Girardin

Technologies de l'internet

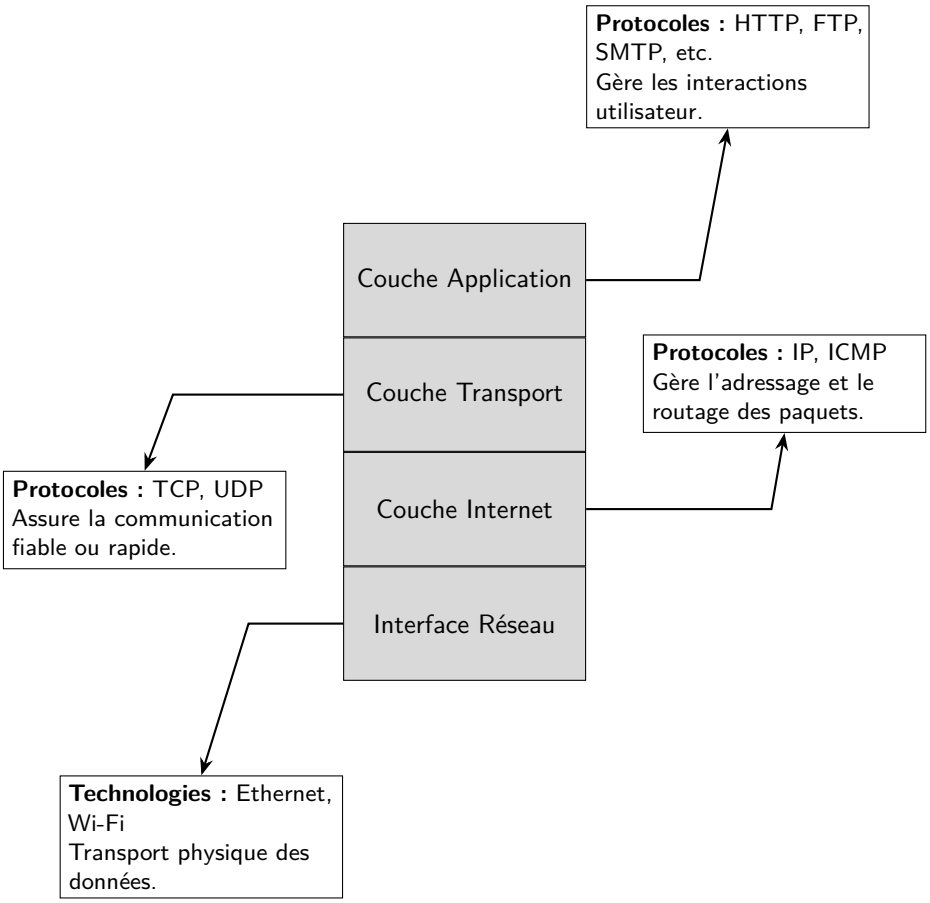
NOTES DE COURS

Département d'informatique et de Recherche Opérationnelle

Montréal 2025
16 janvier 2025

Table des matières

1 Modèle TPC-IP



Définition 1.1 (Modèle TCP/IP)**.** Le modèle TCP/IP est une architecture simplifiée en **4 couches** qui décrit comment les données sont échangées sur un réseau, notamment l'Internet, en décrivant les technologies nécessaires à l'acheminement des données.

1.1 Couche Application

Cette couche inclut des protocoles tels que `HTTP`, `FTP`, `SSH`, `Telnet`, `SMTP`, etc. Elle regroupe les fonctionnalités des couches *Application*, *Présentation*, et *Session* du modèle OSI.

Note

Le Modèle OSI (*Open System Interconnection*) est un modèle en **sept couches** qui décrit les fonctionnalités nécessaires à la communication des systèmes informatiques en réseau

1.2 Couche Transport

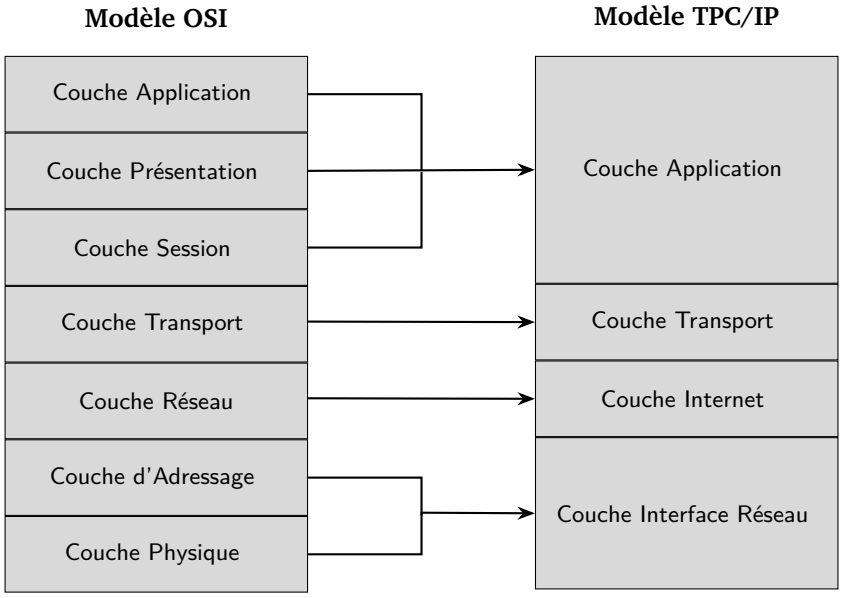
Elle est responsable de la **communication de bout en bout** entre les applications sur différents hôtes. Les protocoles principaux sont `TCP` (*Transmission Control Protocol*) et `UDP` (*User Datagram Protocol*).

1.3 Couche Internet

Responsable de l'**adressage**, du **routage** et de l'**envoi** des paquets entre les réseaux. Les protocoles inclus sont `IP` (*Internet Protocol*) et `ICMP` (*Internet Control Message Protocol*).

1.4 Couche Interface Réseau

Correspond à la liaison de données et à la **couche physique** du modèle OSI. Elle s'occupe de la transmission physique des données à travers le réseau, par exemple via `Ethernet`.



1.5 DARPA

Fondé en 1969, DARPA est l'organisme à l'orgine du premier **réseau à commutation de paquets** : DARPANET. La première démonstration du DARPANET remonte à 1972, mais c'est seulement en 1983, que DARPA adopte la suite de protocoles TPC/IP qui sera la base de la communication Internet.

1.6 Port

Concept 1.2 (Port de Communication)**.** Un **port de communication** est une sorte de point d'accès logique qu'un ordinateur ou un serveur utilise pour identifier un service ou une application spécifique avec laquelle communiquer.

1.6.1 Isolation des Services

Chaque service ou protocole écoute sur un port spécifique, ce qui permet à plusieurs services de fonctionner simultanément sur la même machine **sans interférer**.

1.6.2 Gestion des Connexions Multiples

Un serveur peut gérer plusieurs connexions en utilisant des **combinaisons uniques** de IP et port pour chaque client.

1.6.3 Sécurité

Les ports peuvent être **ouverts ou fermés** par des pare-feux pour limiter l'accès. Connaître les ports associés à des services permet de configurer des règles de sécurité réseau.

PORTS PAR DÉFAUT		
Protocole	**Port**	**Description**
FTP	21	File Transfer Protocol
SMTP	25	Simple Mail Transfer Protocol
HTTP	80	Hyper Text Transfert Protocol
POP	110	Post Office Protocol
IMAP	143	Internet Messagew Access Protocol
Telnet	23	Connexion à une autre machine
Finger	79	Récupération d'info. utilisateur

2 Protocoles Courants

2.1 SSH

Protocole 2.1 (SSH). Permet de se connecter et de contrôler un autre ordinateur via le réseau ; souvent utilisé pour la **connexion à distance**.

`ssh` (connexion) `username` (nom d'u./machine) `@remote_host` (adr. IP) ⟶ `ssh username@remote_host`

Exemple 1 Connexion aux Serveurs du DIRO

```
ssh arcade.iro.umontreal.ca
```

Note

Après avoir établit une connexion sur `arcade` (DIRO), la commande `finger` affiche le contenu des fichier `.plan` et `.project` du `~/home/`

2.2 FTP

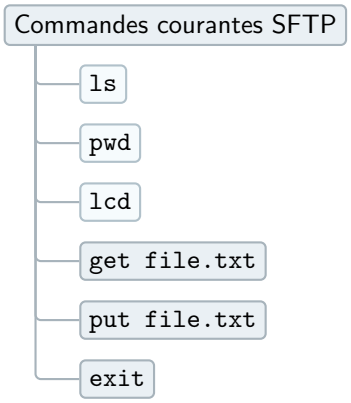
Protocole 2.2 (SFPT). Permet d'enclencher un mécanisme de connexion à distance pour effectuer le **transfert sécuritaire de données**

DIFFÉRENCES ENTRE FPT ET SFTP		
Caractéristique	**FTP**	**SFTP**
Sécurité	Non sécurisé	Sécurisé grâce à SSH
Port par défaut	21	22
Transfert de données	Non chiffré	Chiffré
Protocole sous-jacents	Protocole FTP	Protocoles FTP, SSH

`sftp` (connexion) `username` (nom d'u./machine) `@remote_host` (adr. IP) ⟶ `sftp username@remote_host`

Exemple 2 Transfert Sécuritaire de Fichiers sur les Serveurs du DIRO

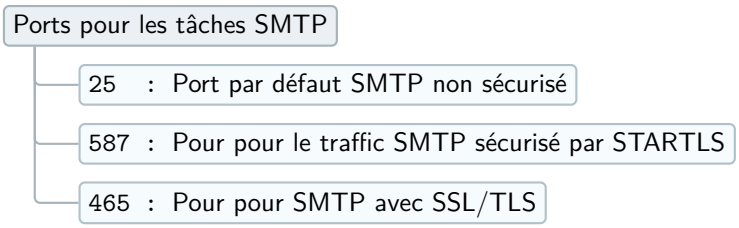
```
sftp arcade.iro.umontreal.ca
```



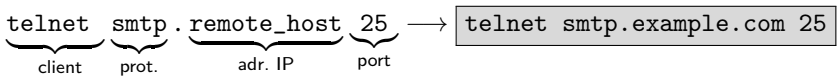
2.3 SMTP

Protocole 2.3 (SMTP). Permet d'**envoyer des e-mails** dun client de messagerie (comme `Outlook` ou `Thunderbird`) à un serveur de messagerie, ou entre serveurs de messagerie.

`SMTP` utilise le protocole `TCP` (*Transmission control Protocol*) et peut être utilisé sur différents ports, dépendamment du besoin.



`SMTP` peut invoqué par différents clients tels que `telnet`, `OpenSSL` et `netcat`, puisque `SMTP` repose sur `TCP` et que ces trois clients **prennent en charge** `TCP`.



Exemple 3 Utilisation de SMTP via `netcat`

```
nc -v mail.iro.umontreal.ca 25

        HELO mydomain.com
        MAIL FROM:<your_email@example.com>
        RCPT TO:<recipient@example.com>
        DATA
        Subject: Test Email with Netcat
```

```
Hello! This email was sent using SMTP and netcat.
.
QUIT
```

Détails des options

2.4 STMP via l'outil curl

Définition 2.4 (Commande `curl`). L'outil `curl` est polyvalent et permet de **gérer différents protocoles** en combinant des séquences de commandes.

Exemple 4 Utilisation de `curl` pour envoyer un e-mail

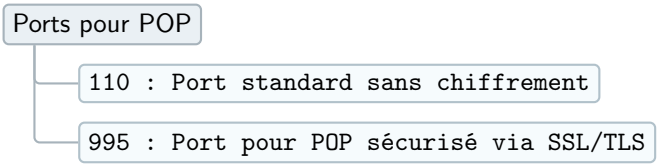
```
curl --url 'smtps://mail.iro.umontreal.ca:465' \
     --ssl-reqd                                \
     --mail-from 'felipe@iro.umontreal.ca'     \
     --mail-rcpt 'felipe@iro.umontreal.ca'     \
     --upload-file mail.txt                    \
     --user 'felipe@iro.umontreal.ca:passwd'   \
     --insecure
```

Détails des options

2.5 POP

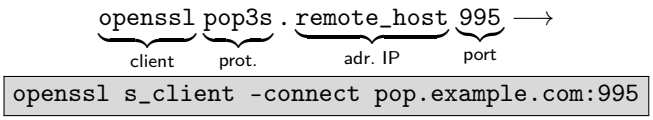
Protocole 2.5 (POP). Permet de **récupérer des e-mails** depuis un serveur de messagerie distant jusqu'à un client de messagerie (comme `Outlook` ou `Thunderbird`), en téléchargeant les messages localement. Le protocole permet aussi de **supprimer tous les messages** lus.

`POP` utilise le protocole `TCP` (*Transmission Control Protocol*) et fonctionne généralement sur deux ports standards, selon le niveau de sécurité.



Contrairement à IMAP, POP télécharge les messages et les supprime par défaut du serveur, bien que cette option puisse être configurée dans certains clients.

POP peut être invoqué par des outils comme telnet et openssl, car il repose également sur TCP. Ces outils permettent de tester la connectivité et d'interagir manuellement avec un serveur POP.



Exemple 5 Utilisation de POP via telnet

```
telnet pop.example.com 110
+OK POP3 server ready
USER your_email@example.com
+OK User accepted
PASS your_password
+OK Password accepted
LIST
+OK 2 messages
1 1024
2 2048
RETR 1
+OK Message follows
Subject: Test Email
Hello! This email was retrieved using POP and telnet.
.
QUIT
+OK Goodbye
```

Détails des options

2.6 POP via l'outil curl

Exemple 6 Utilisation de pop3 via curl

```
curl -o mail.pop -v --ssl-reqd -u 'felipe:passwd' \
                    --request UIDL                \
```

```
--url pop3://mail.iro.umontreal.ca
```

Détails des options

2.7 IMAP

Protocole 2.6 (IMAP). Permet de **gérer des e-mails à distance** depuis un serveur de messagerie tout en maintenant les messages sur le serveur. Contrairement à POP, IMAP synchronise les actions entre le client et le serveur (e.g., marquer un e-mail comme lu).

IMAP utilise le protocole TCP et fonctionne sur des ports standards.

- Ports pour IMAP
  - 143 : IMAP sans chiffrement
  - 993 : IMAP avec SSL/TLS

IMAP peut être invoqué par des outils comme curl, offrant une interface scriptable pour interagir avec les boîtes de réception distantes.

Exemple 7 Connexion simple via curl avec IMAP

```
curl --insecure \
     --url 'imaps://mail.iro.umontreal.ca' \
     --user 'felipe:passwd'
```

Détails des options

Exemple 8 Examiner une boîte de réception avec EXAMINE INBOX

```
curl --insecure \
     --url 'imaps://mail.iro.umontreal.ca' \
     --user 'felipe:passwd' \
     --request "EXAMINE␣INBOX"
```

Détails des options

Exemple 9 Télécharger un message avec `UID`

```
curl -o mail.imap \
     --insecure \
     --url 'imaps://mail.iro.umontreal.ca/INBOX;UID=14050' \
     --user 'felipe:passwd'
```

Détails des options

Exemple 10 Afficher l'en-tête d'un e-mail avec `head`

```
head -n 8 mail.imap
```

Détails des options

Exemple 11 Afficher la fin d'un e-mail avec `tail`

```
tail -n 20 mail.imap
```

Détails des options

3 Internet et HTTP

3.1 Fonctionnement du Web et URI

Concept 3.1 (Fonctionnement du Web). Le Web est la conséquence de l'interaction entre **trois ensembles de technologies** , soient les `URI` (*Universal Ressource Locator*), le protocole `HTTP`, et le langage `HTML`

$$\text{Web} = \texttt{URI} + \texttt{HTTP} + \texttt{HTML}$$

Une `URI` fait référence au **nom général** de la ressource, alors que l'`URL` permet de **localiser spécifiquement** la ressource et la portion de la ressource qu'on désire consulter.

Concept 3.2. Une URL est peut être composée, entre autres, d'un **schéma** (p. ex `http`), d'un **nom de domaine**, d'un **chemin**, de **paramètres** et d'une **ancre**.

$$\underbrace{\texttt{scheme}}_{\text{protocol}}\texttt{:[}\ \underbrace{\texttt{//authority}}_{\text{authentification ou domaine}}\ \texttt{]}\underbrace{\texttt{path}}_{\text{chemin}}\texttt{[}\underbrace{\texttt{?query}}_{\text{paramètres}}\texttt{][}\underbrace{\texttt{\#anchor}}_{\text{ancre}}\texttt{]}$$

Exemple 12 URL simple

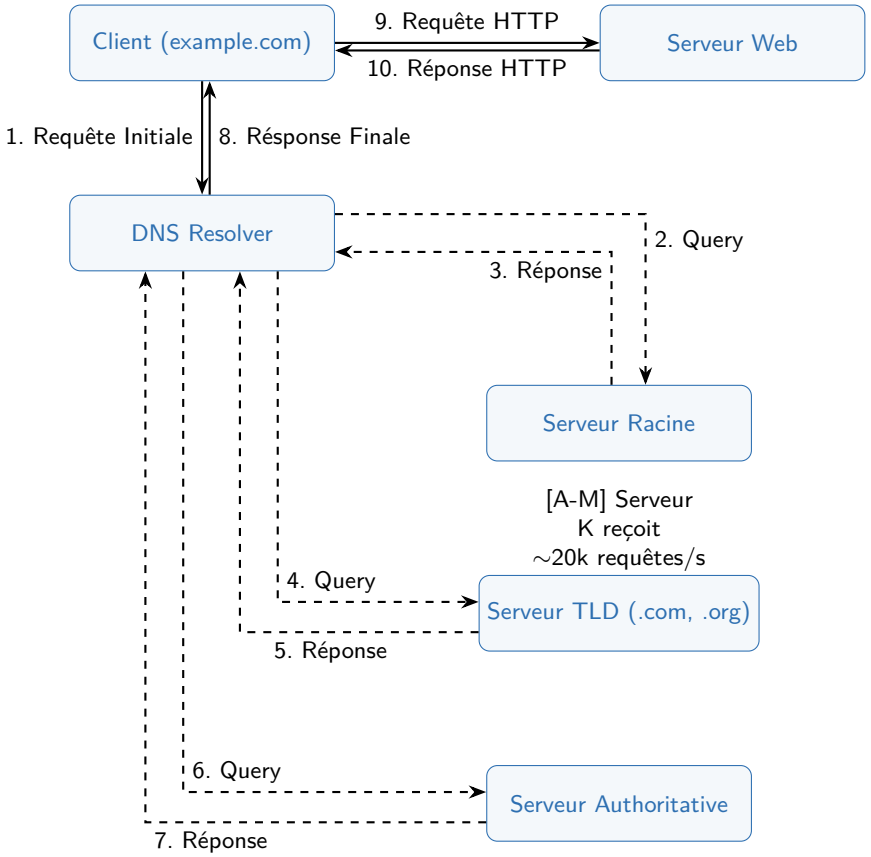
$$\texttt{http://}\underbrace{\texttt{www.iro.umontreal.ca}}_{\text{nom de domaine}}\texttt{/}\underbrace{\texttt{\~felipe/new-home/frontal.php}}_{\text{path}}\texttt{?}\underbrace{\texttt{page=resume}}_{\text{query}}$$

Concept 3.3 (Encodage URI). L'encodage d'un `URI` repose sur un **remplacement des caractères non valides** ou réservés par un code hexadécimal précédé du caractère

Caractère	Signification	Encodage
%	Pourcentage (réservé)	%25
(espace)	Espace	%20
#	Fragment (ancre)	%23
&	Délimiteur de paramètres	%26
'	Apostrophe	%27
/	Séparateur de chemin	%2F
?	Début de la query string	%3F
=	Affectation dans une query	%3D
+	Addition ou espace	%2B

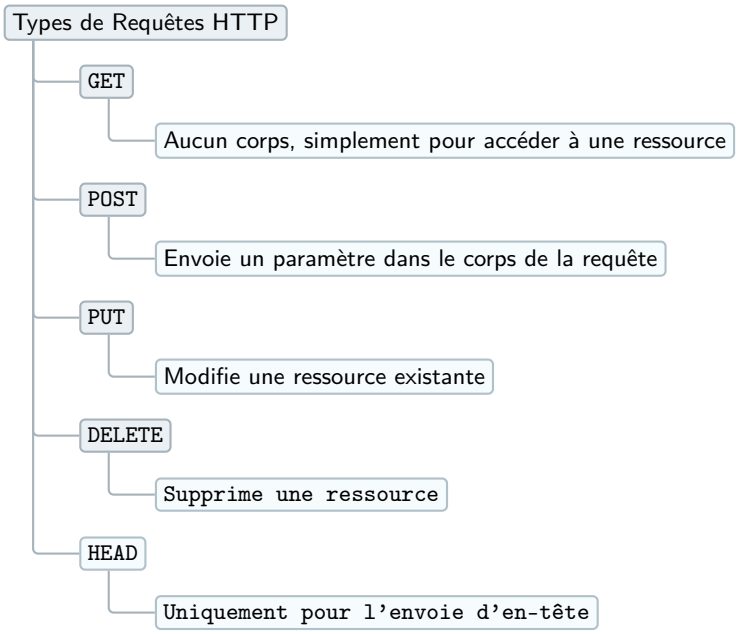
3.2 DNS

Concept 3.4 (Domain Name System). Le DNS est un système distribué utilisé pour **traduire les noms de domaine** lisibles par l'humain en adresses IP compréhensibles par les machines.



3.3 HTTP

Concept 3.5 (HyperText Transfer Protocol (HTTP))**.** Le **HTTP** est un protocole de communication utilisé pour **transférer des données sur le Web**. Il est basé sur un modèle client-serveur où le client (p. ex., un navigateur web) envoie une requête au serveur, et le serveur répond avec les ressources demandées. Plusieurs **proxy entre un client et un serveur peuvent intervenir et certains proxy peuvent mémoriser les réponse (ISP)**. Ainsi, les fournisseurs de services internet peuvent déterminer l'empreinte digitale de chacun de leurs utilisateurs en fonctions des sites que ceux-ci visitent gardés en mémoire cache par l'ISP.



3.3.1 Syntaxe d'une requête HTTP

Note

Chaque requête HTTP est **indépendante des autres**, ce qui simplifie les interactions mais nécessite des mécanismes supplémentaires (comme les cookies) pour maintenir l'état.

3.4 Utilisation de `curl` pour requêtes HTTP

Exemple 13 Requête `GET` simple avec `curl`

```
curl https://www.google.ca/
```

L'outil `curl` est utilisé pour faire une requête `GET` qui contient uniquement une en-tête.

Détails de la requête `GET` simple

Exemple 14 Requête `GET` détaillée (`-v`) avec `curl`

```
curl -v https://www.google.ca/
```

Cette commande engendre une requête qui demande la page racine (`/`) du site `www.google.ca` via HTTP 1.1. Le `User-Agent` spécifie que `curl` est utilisé. `Accept` indique que tous les types de contenu (`*/*`) sont acceptés.

```
GET / HTTP/1.1
Host:www.google.ca
User-Agent curl/7.58.0
Accept:*/*
```

Détails de la requête `GET` détaillée

Exemple 15 Télécharger une page web avec `-o` : réponse *moved permanently*

```
curl -o out -v https://diro.umontreal.ca/accueil/
```

Cette commande `curl` effectue une requête GET vers l'**ancienne page d'acceuil** du DIRO avec des options. L'option `-o out` sauvegarde la réponse dans un fichier nommé `out`. L'option `-v` : Affiche les détails de la requête et de la réponse qui indique que **l'url a été déplacé**.

Détails de la réponse

Exemple 16 Redirection temporaire avec `curl` : *moved permanently & temporary redirect*

```
curl -v http://diro.umontreal.ca
```

Cette commande `curl` effectue une requête `GET` vers l'**ancienne URL** du DIRO. La réponse montre une redirection initiale permanente vers une version HTTPS de l'URL, suivie d'une redirection temporaire vers la page d'accueil.

Détails des réponses

Exemple 17 Télécharger des données JSON

```
curl -o x.json -v https://jsonplaceholder.typicode.com/users
```

Détails des options

Exemple 18 Envoyer des données via un fichier avec `-d`

```
curl -v -d @x.json https://jsonplaceholder.typicode.com/posts
```

Détails des options

Exemple 19 Envoi d'une requête `POST` avec des données JSON

```
curl -X POST -d  x.jons                          \
     https://jsonplaceholder.typicode.com/posts \
     -H 'Content-Type:␣application/json'
```

Fichier `x.json` :

```
{
    "title": "foo",
    "body": "bar",
    "userId": 1
}
```

Détails des options

Exemple 20 Modification de données avec une requête `PUT`

```
curl -X PUT -v -d                                          \
'{"id":␣101,␣"title":␣"foo␣h","body":␣"bar","userId":␣1}' \
     https://jsonplaceholder.typicode.com/posts/1       \
     -H 'Content-Type:␣application/json'
```

Détails des options

h

Appendices

A.1 Protocoles Courants (Détails des Commandes UNIX)

A.1.1 Utilisation de SMTP via `netcat`

`HELLO` Identifie le client auprès du serveur `SMTP`

`MAIL FROM` Indique l'adresse e-mail de l'expéditeur

`RCP TO` Indique l'adresse email du destinataire

`DATA` Permet de saisir le contenu du message

`.` Termine le message

`QUIT` Termine la session.

A.1.2 Utilisation de curl pour envoyer un e-mail

`-url` spécifie l'URL du serveur `SMTP`

`-ssl-reqd` force l'utilisation de `SSL/TLS` ;

`-mail-from` indique l'adresse e-mail de l'expéditeur

`-mail-rcpt` spécifie l'adresse e-mail du destinataire ;

`-upload-file` fournit le contenu du message

`-user` inclut les informations d'identification pour l'authentification sur le serveur `SMTP` ;

`-insecure` désactive la vérification du certificat SSL/TLS.

A.1.3 Utiliation de POP via telnet

`USER` spécifie l'adresse e-mail de l'utilisateur ;

`PASS` fournit le mot de passe pour l'authentification ;

`LIST` affiche la liste des messages disponibles avec leur taille ;

`RETR` télécharge un message spécifique depuis le serveur ;

`DELE` suprimme un email en fournissant **le numéro**

`TOP` affiche les lignes d'un message

`QUIT` termine la session.

A.1.4 Utilisation de pop3 via curl

`-o mail.pop` spécifie que la sortie (le contenu des messages) doit être écrite dans un fichier nommé `mail.pop` ;

`-v` active le mode *verbose*, qui affiche des informations détaillées sur la connexion et le transfert des données, utile pour le débogage ;

`-ssl-reqd` force l'utilisation de `SSL/TLS` pour sécuriser la connexion au serveur POP ;

`-u 'felipe:passwd'` fournit les informations d'identification pour s'authentifier auprès du serveur POP ;
`-request UIDL` envoie la commande UIDL au serveur, qui renvoie une liste des messages présents dans la boîte de réception avec leurs identifiants uniques, sans les télécharger ;
`-url pop3://mail.iro.umontreal.ca` spécifie l'URL du serveur POP, permettant de se connecter au serveur à l'adresse `mail.iro.umontreal.ca`.

A.1.5 Connexion simple via `curl` avec `IMAP`

`-url` spécifie l'URL du serveur IMAP pour établir une connexion ;
`-insecure` désactive la vérification des certificats SSL/TLS (à éviter en production) ;
`-user` fournit les informations d'identification pour s'authentifier auprès du serveur.

A.1.6 Examiner une boîte de réception avec `EXAMINE INBOX`

`-request` permet d'envoyer une commande IMAP spécifique, comme `EXAMINE INBOX`, pour afficher des informations détaillées sur la boîte de réception.

A.1.7 Télécharger un message avec `UID`

`-o` enregistre le contenu du message dans un fichier local (e.g., `mail.imap`) ;
`UID` identifie un message spécifique dans la boîte de réception pour téléchargement.

A.1.8 Afficher l'en-tête d'un e-mail avec `head`

`head -n` affiche les premières lignes du fichier contenant le message pour examiner l'en-tête.

A.1.9 Afficher la fin d'un e-mail avec `tail`

`tail -n` affiche les dernières lignes du fichier pour examiner la fin du message (e.g., signature ou pièces jointes encodées).

A.2 Détails des commandes `curl` pour requêtes `GET`

A.2.1 Détail de la requête `GET` simple avec `curl`

`curl` envoie une requête GET par défaut à l'URL spécifiée et affiche la réponse brute dans le terminal.

A.2.2 Détail de la requête `GET` détaillée avec `curl`

`-v` active le mode verbose pour afficher les détail (e.g., en-têtes HTTP).

A.2.3 Détail de la réponse *Moved Permanently*

`Pour la réponse suivante`

```
HTTP/1.1 301 Moved Permanently
Date: Tue, 07 Jan 2020 03:40:45 GMT
Server: Apache
Location: http://diro.umontreal.ca/
Content-Length: 351
Content-Type: text/html; charset=iso-8859-1
```

`HTTP/1.1 301 Moved Permanently` indique que la ressource a été déplacée de façon permanente.
`Date` spécifie lheure de la réponse: `Tue, 07 Jan 2020 03:40:45 GMT`.
`Server` indique que le serveur utilisé est `Apache`.
`Location` redirige vers lURL `http://diro.umontreal.ca/`.
`Content-Length` indique que la longueur du contenu est de 351 octets.
`Content-Type` précise que le contenu est du HTML encodé

A.2.4 Détail des réponses *Moved Permanently* et *Temporary Redirect*

`Pour les réponses suivantes`

```
HTTP/1.1 301 Moved Permanently
Location: https://diro.umontreal.ca/
Server: BigIP
Connection: Keep-Alive
Content-Length: 0

HTTP/1.1 307 Temporary Redirect
Date: Mon, 06 Jan 2020 05:50:18 GMT
Vary: Host
Location: https://diro.umontreal.ca/accueil/
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Server-By: web1
Age: 0
grace: none
X-Cache: MISS
Connection: keep-alive
Set-Cookie: BIGipServerCms.dgtic-pool=1076201482.20480.0000; path=/; Httponly; Secure
```

`HTTP/1.1 301 Moved Permanently` indique que la ressource a été déplacée de façon permanente vers `https://diro.umontreal.ca/`.
`Location` redirige vers une version HTTPS de l'URL.
`Server` précise que le serveur utilisé est `BigIP`.
`Connection` spécifie que la connexion reste active (`Keep-Alive`).
`HTTP/1.1 307 Temporary Redirect` indique une redirection temporaire vers `https://diro.umontreal.ca/accueil/`.
`Date` précise lheure de la réponse: `Mon, 06 Jan 2020 05:50:18 GMT`.
`Vary` informe que la réponse varie selon lhôte.

`Set-Cookie` configure un cookie pour des sessions futures, avec des options sécurisées (`Httponly` et `Secure`).

A.2.5 Télécharger des données JSON avec `curl`

`-o` enregistre les données JSON retournées par l'URL dans un fichier (`x.json`).

A.2.6 Envoyer des données via un fichier avec `curl`

`-d` permet d'envoyer des données dans le corps de la requête HTTP. Le symbole `@` indique que les données doivent être lues depuis un fichier (`x.json`).

A.2.7 Requête POST avec des données JSON

`-X POST` spécifie que la requête doit utiliser la méthode `POST`.
`-H 'Content-Type: application/json'` indique que les données envoyées sont en format JSON.

A.2.8 Modification avec une requête PUT

`-X PUT` utilise la méthode `PUT` pour modifier ou remplacer une ressource existante. Le contenu JSON est transmis dans le corps de la requête, avec l'en-tête `-H 'Content-Type: application/json'` pour indiquer le type de données.