# BTS SN

| Période | Sem 1 | Sem2 | Sem3 | Sem4 |
|---|---|---|---|---|
| Volume horaire | Cours/TD | | TP | |
| | 4 | | 11 | |

## Première approche de la POO

## Journal lumineux

## Documents à rendre :

- Un document de synthèse contenant la réponse aux questions des différentes activités. Les diagrammes UML seront ajoutés dans ce document de synthèse.
- Le code source du fichier main.cpp à la fin des activités 2, 3 et 4.

## Ressources :

Les ressources pour ce module sont :

- Les PDF du constructeurs de l'afficheur dans le répertoire documentsConstructeur.
- Les fichiers sources du projet (fichier main.cpp à écrire) dans le répertoire sources de cette archive ;
- Une documentation de l'API afficheur météo au format HTML se trouve dans le répertoire html de cette archive. Ouvrir le fichier html/index.html dans un navigateur pour afficher la page d'accueil.

## Matériels

### En salle A108

**Poste de l'afficheur - 172.20.182.10**

- Pour SSH et FTP, utilisez le compte suivant :
  - Utilisateur : etudiant
  - Mot de passe : etudiant
- Pour le serveur UDP
  - Port du serveur UDP : 4321

# Activité 1 - Présentation du cahier des charges et analyse (30 min)

## Indicateur temporel

| Questions | 0h15 | | 0h30 | |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

## Présentation du cahier des charges

Un responsable d'une agence d'assurance désire améliorer l'accueil de ses visiteurs. Il possède un afficheur à LED, mais les informations qu'il affiche ne sont pas dynamiques. Il désire que l'hôtesse d'accueil puisse ajouter, supprimer ou modifier aisément des messages sur l'afficheur. De plus, il souhaiterait afficher les conditions météorologiques actuelles ainsi que les prévisions.

## Analyse du cahier des charges

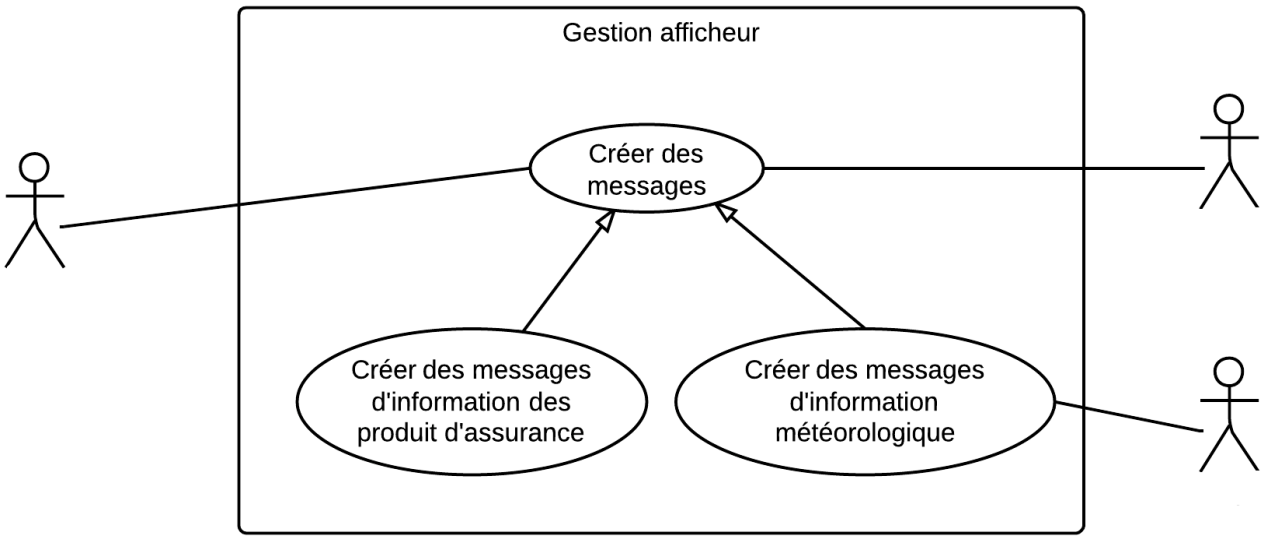
L'objectif de cette partie est, à partir du cahier des charges précédent, de mettre en place les diagrammes UML de cas d'utilisation et de déploiement de notre système.

### Le diagramme de cas d'utilisation

Le diagramme de cas d'utilisation présente la liste exhaustive des acteurs interagissant avec notre système. Dans notre cas, notre système est l'application demandée par le gérant d'une agence assurance. La seule personne interagissant avec cette application sera l'hôtesse d'accueil. Par contre, cette application devra communiquer avec un afficheur à LED déjà existant et un serveur web météorologique.

1. Complétez le diagramme de cas d'utilisation suivant en ajoutant le nom des acteurs :
   - l'hôtesse d'accueil ;
   - le client
   - le serveur web météorologique

   Saisissez ce diagramme avec votre logiciel d'édition de diagramme UML.



## Le diagramme de déploiement

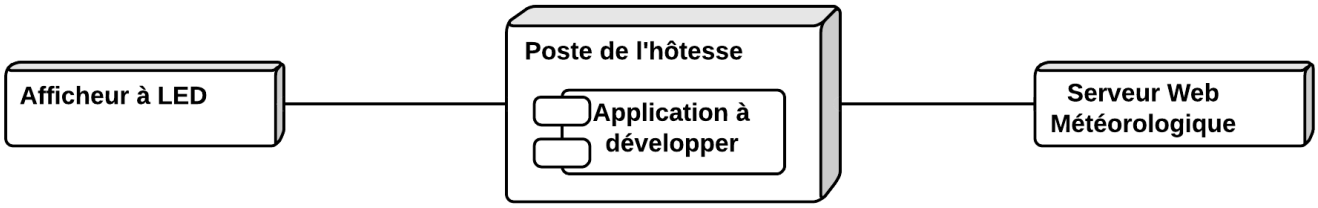
Le logiciel informatique à mettre en place sera utilisé par l'hôtesse d'accueil. C'est là notre mission. Notre logiciel doit :

- communiquer avec l'afficheur à LED ;
- communiquer avec un serveur web météorologique sur internet.

2. En regardant l'afficheur ou en consultant la documentation technique du constructeur de l'afficheur, indiquez comment sera connecté l'afficheur au poste informatique de l'hôtesse d'accueil.
3. En utilisant un moteur de recherche, trouvez le nom du protocole que devra utiliser notre application pour communiquer avec le serveur web météorologique.

Les informations que vous venez de trouver sont essentielles pour constituer le diagramme de déploiement.

4. Complétez maintenant le diagramme de déploiement. Saisissez-le avec votre logiciel d'édition de diagramme UML.



# Activité 2A - Découverte de l'existant : le logiciel fourni - Windows (1h30)

## Indicateur temporel

| Questions | 0h30 | | 1h00 | | 1h30 | |
|---|---|---|---|---|---|---|
| 1-3 | | | | | | |
| 4 | | | | | | |
| 5-7 | | | | | | |
| 8-10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |

## Introduction

Comme vous l'avez compris dans le cahier des charges, le constructeur de l'afficheur fourni un logiciel peu intuitif. Voyons comment fonctionne ce logiciel. Ce logiciel s'appelle 'New Sign'. Cette partie vous aidera à le prendre en main et à le paramétrer.

## Installation et paramétrage du logiciel New Sign

1. Installez le logiciel *New Sign* (si ce n'est pas déjà fait).
2. Connectez l'afficheur sur un port USB de votre poste.
3. Dans le gestionnaire de périphériques, recherchez le numéro du port série virtuel attribué à l'afficheur. Indiquez ce numéro dans le logiciel New Sign.
4. En utilisant votre moteur de recherche favori, expliquer ce qu'est un port série virtuel.

Maintenant que le logiciel est correctement paramétré, vous pouvez modifier le texte à afficher.

5. Affichez votre nom et prénom sur l'afficheur.
6. Changez les effets entrant et sortant de votre texte.
7. Affichez la date et l'heure.

## Analyse des trames échangées entre le PC et l'afficheur (1h)

Nous allons maintenant lancer un logiciel qui va espionner les données qui sont transmises sur le port série virtuel entre le logiciel New Sign et notre afficheur. Ce logiciel s'appelle AspiCom.

8. Lancez le logiciel AspiCom. Indiquez les paramètres suivants :
   - Communication : Numéro du port COM virtuel et Mode transparent.
   - Configuration : 9600 baud, 8 bits, pas de parité, 1 bit stop.

Maintenant que le logiciel d'espionnage est correctement paramétré, vous pouvez analyser les données transmises à l'afficheur.

9. Envoyez à l'afficheur le message court "hello".
10. Écrivez sur 3 lignes les 3 messages envoyés comme dans l'exemple ci-dessous (l'exemple permet d'afficher "BTS IRIS" sur l'afficheur) :

```
<ID00><BE>05<E>
<ID00><L1><PA><FE><MA><WC><FE>BTS IRIS0C<E>
<ID00><BF>06<E>
```

Le premier et le 3ème message sont des formules de politesse en langage afficheur (du type "Bonjour" et "Au revoir" pour les êtres sociables).

Dans la suite de ce projet, nous nous intéresserons uniquement au 2ème message transmis à l'afficheur. Ce 2ème message se décompose comme suit :

```
<ID00>      <L1><PA><FE><MA><WC><FE>BTS SN      0C      <E>

Identifiant de l'afficheur      Bloc de données : Les effets d'affichage puis le texte à afficher      Checksum      Fin
```

11. En utilisant la documentation du constructeur de l'afficheur (590996-da-01-en-Communication_protocol_LED_Displ_Board.pdf, pages 5 à 8), expliquez la signification de <L1>, <PA>, <FE>, <MA>, <WC> et <FE>.
12. Vérifier que le checksum est correct en effectuant un OU EXCLUSIF entre tous les caractères du bloc de données.

Une fois cette première partie du mini-projet terminée, vous connaissez :

- la configuration du port série virtuel de l'afficheur ;
- la structure de la trame à envoyer pour afficher une ligne.

## Activité 2B - Découverte de l'existant : envoi de la trame (1h30)

### Indicateur temporel

| Questions | 0h30 | | 1h00 | | 1h30 | |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2,3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |

### Objectifs

Dans cette activité de découverte du matériel, nous n'utiliserons pas le logiciel fourni avec l'afficheur : il ne fonctionne pas sur la distribution Ubuntu. Par contre, nous découvrirons comment installer le matériel pour cette distribution et nous verrons comment envoyer simplement un message à l'afficheur en suivant la documentation du constructeur.

### Ouverture d'un terminal distant

Vous allez tous travailler sur le même afficheur. Il est connecté en USB sur une machine disposant d'un système d'exploitation Ubuntu.

Pour travailler à distance sur l'afficheur, vous allez ouvrir un terminal distant (sur votre machine) de

### Installation du matériel sur une distribution Linux

Vous trouverez une documentation d'installation de l'afficheur pour un OS Linux dans la page d'accueil de la documentation.

1. Allez au sous-titre 'Quel Port série ?' de la documentation HTML de ce projet.
   Saisissez la commande indiquée et donnez le nom du port série utilisé pour l'afficheur.

### Écriture sur le port série

2. À l'aide de la commande ssh, ouvrez un terminal distant du poste afficheur (voir l'annexe 4)

Une fois le terminal du poste de l'afficheur ouvert, vous allez utiliser la commande send2LedDisplay pour envoyer un message à l'afficheur (cette commande n'est pas une commande Unix, elle a été mise au point pour l'occasion).

```
send2LedDisplay /dev/ttyUSB0 '<ID00><L1><PA><FE><MA><WC><FE>BTS SN0C<E>'
```

La commande précédente envoie sur le port /dev/ttyUSB0 le message « <ID00><L1><PA><FE><MA><WC><FE>BTS SN0C<E> ».

Le message à envoyer à l'afficheur doit respecter un format prédéfini par le constructeur de ce matériel. Dans le tableau suivant, vous trouverez une description du message :

| <ID00> | <L1><PA><FE><MA><WC><FE>BTS SN | 0C | <E> |
|---|---|---|---|
| Identifiant de l'afficheur | Bloc de données : Les effets d'affichage puis le texte à afficher | Checksum | Fin |

3. Dans le terminal distant du poste afficheur, en utilisant la commande send2LedDisplay, envoyer ce message à l'afficheur. Vérifiez que le message s'affiche correctement. Pour

différencier votre message des autres, vous pouvez ajoutez 2 fois la même lettre dans le texte à afficher par exemple : « <ID00><L1><PA><FE><MA><WC><FE>BTS SNGG0C<E> ». J'ai ajouté ici deux lettres G majuscules.

4. En utilisant la documentation du constructeur de l'afficheur (590996-da-01-en-Communication_protocol_LED_Displ_Board.pdf, pages 5 à 8), expliquez la signification de <L1>, <PA>, <FE>, <MA>, <WC> et <FE>.
5. Vérifier que le checksum est correct en effectuant un OU EXCLUSIF entre tous les caractères du bloc de données.
6. Sans modifier les effets d'affichage, affichez le texte : « Bienvenue en BTS SN ». Écrire la trame à envoyer dans votre compte-rendu et expliquez comment vous l'avez trouvé. Détaillez le calcul du checksum.
7. Envoyez votre message à l'afficheur. Vérifiez ainsi que votre message est correct.

Une fois cette première partie du mini-projet terminée, vous connaissez :

- la configuration du port série virtuel de l'afficheur ;
- la structure de la trame à envoyer pour afficher une ligne.

## Activité 3A - Affichage du nom et du prénom avec la classe Afficheur (3h)

Dans cette partie, vous allez développer une application en mode console qui devra afficher votre nom et votre prénom sur l'afficheur.

Deux étapes seront nécessaires :

- écrire le programme principal.
- compiler et exécuter votre programme sur une machine distante.

### Analyse de la classe Afficheur

Dans cette partie, vous prendrez en main la documentation de la classe Afficheur. Cette documentation a été automatiquement générée à partir des commentaires des fichiers sources. L'outil utilisé s'appelle Doxygen.

1. Dans le répertoire /html/ de l'archive ce module, vous trouverez un fichier nommé index.html. Il s'agit de la page d'accueil de la documentation. Ouvrez cette page avec votre navigateur web.
2. Dans l'onglet Classes, cliquez sur la classe Afficheur.
3. D'après le premier paragraphe du sous-titre « Description détaillée » de cette classe, expliquez dans vos propres termes l'utilité de la classe Afficheur.

Les paragraphes suivants de la documentation expliquent comment utiliser cette classe dans un programme.

### Prise en main de l'application

La prise en place de l'application est décrite dans l'annexe 2A (jusqu'au sous-titre : Édition des fichiers sources) de ce document.

4. En utilisant l'annexe 2A, éditez le fichier main.cpp.

### Écriture du programme principal : le fichier main.cpp

Dans cette partie, vous allez modifier exclusivement le fichier main.cpp de votre répertoire sources.

#### Les inclusions

Puisque vous allez utilisez la classe Afficheur, vous devez inclure le fichier Afficheur.h qui vous a été fourni. Cette inclusion ce fait comme suit :

```
#include "Afficheur.h"
```

5. Ajoutez l'inclusion précédente dans le fichier *main.cpp*.

#### La zone d'ajout de code

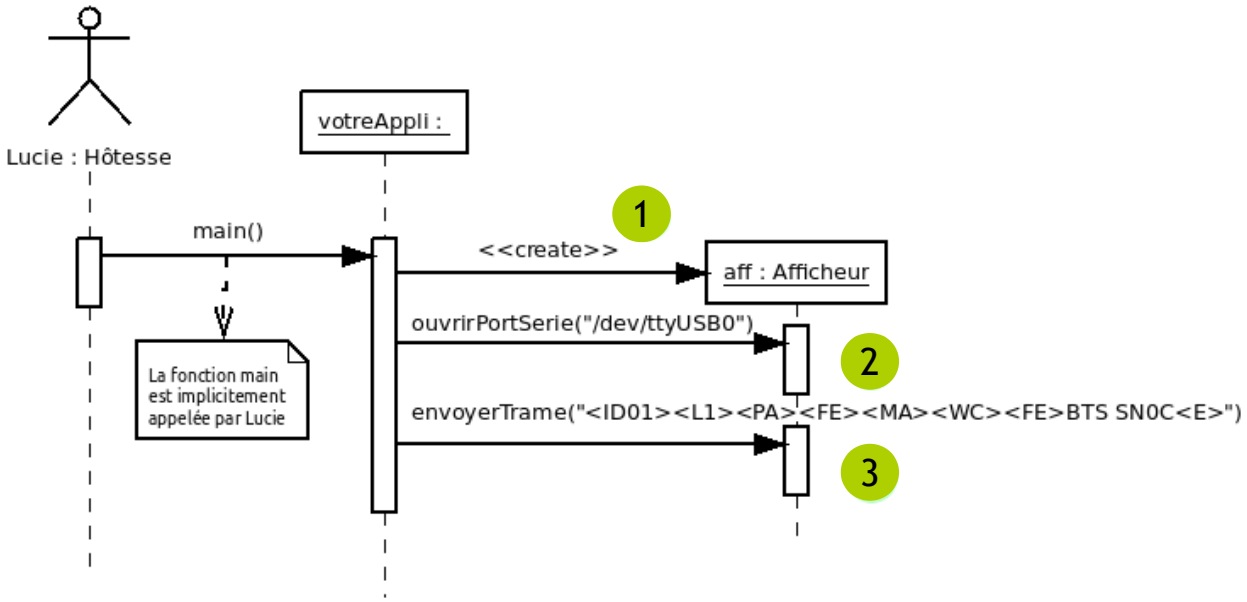
Le code sera à insérer dans la fonction *main()*, en dessous du commentaire « Votre code est à ajouter ici : » comme indiqué ci-dessous :

```
int main()
{
      // Votre code est à ajouter ici :

      return 0 ;
}
```

#### Description du programme principal par un diagramme de séquence

Le diagramme de séquence suivant vise à expliquer le contenu de la fonction *main()* :




Trois étapes apparaissent clairement dans ce diagramme :

1. créer un objet de classe Afficheur ;
2. appeler la méthode *ouvrirPortSerie()* pour indiquer le nom du port série utilisé par notre afficheur ;
3. envoyer un message grâce à la méthode *envoyerTrame()*.

## 1 La création d'un objet de la classe Afficheur

Dans le diagramme de séquence, la création de l'objet se fait par la flèche <<create>>. Il y a alors appel au constructeur *Afficheur()* de la classe Afficheur.

6. En utilisant votre cours, rappelez l'utilité d'un constructeur.
7. En utilisant le 2ème paragraphe du sous-titre « Description détaillée » de la documentation de la classe Afficheur, écrire l'instruction permettant de créer un objet nommé aff de la classe Afficheur.

## 2 Paramétrage du port série

Dans le diagramme de séquence, la seconde étape est l'appel de la méthode *ouvrirPortSerie()*.

8. En utilisant la documentation de la classe Afficheur, expliquez le rôle de la méthode *ouvrirPortSerie()*.
9. En utilisant le 3ème paragraphe du sous-titre « Description détaillée » de la documentation de la classe Afficheur, écrire l'instruction permettant d'appeler la méthode *ouvrirPortSerie()* de l'objet aff. Le port série utilisé est /dev/ttyUSB0.

## 3 Envoi d'une trame

Dans le diagramme de séquence, la dernière étape est l'appel de la méthode *envoyerTrame()*.

10. En utilisant la documentation de la classe Afficheur, expliquez le rôle de la méthode *envoyerTrame()*.
11. En utilisant le 5ème paragraphe du sous-titre « Description détaillée » de la documentation de la classe Afficheur, écrire l'instruction permettant d'appeler la méthode *envoyerTrame()* de l'objet aff. Le message à transmettre sera celui du diagramme de séquence ci-dessus.

## Compilation et exécution

Il est maintenant temps de compiler votre programme et de le tester.

12. Compilez et exécutez votre application en utilisant l'annexe 3A.

## Affichage de votre nom et de votre prénom

Maintenant que vous êtes capable d'afficher un message donné sur l'afficheur, vous allez affichez votre nom et votre prénom.

13. Écrire la trame à envoyer en suivant les recommandations suivantes :
• laissez les effets d'affichage aux mêmes valeurs.
• modifiez le texte à afficher
• calculez le checksum (voir l'activité 1)
14. Modifiez maintenant votre code source avec votre nouvelle trame. Ensuite, compilez et exécutez votre application.

## Modification des effets d'affichage

On désire maintenant dans cette partie modifier quelques effets d'affichage :

- l'apparition du texte se fera avec un effet neige ;
- le texte sera clignotant;
- le temps d'affichage sera de 20 secondes
- le texte disparaîtra sans effet.

15. Écrire la trame à envoyer pour respecter ces effets. Vous utiliserez la documentation du constructeur pour trouver les valeurs.
16. Modifiez votre code source avec votre nouvelle trame. Puis compilez et exécutez votre application.

## Activité 4A - Modifier le texte et les effets d'affichage avec la classe Ligne

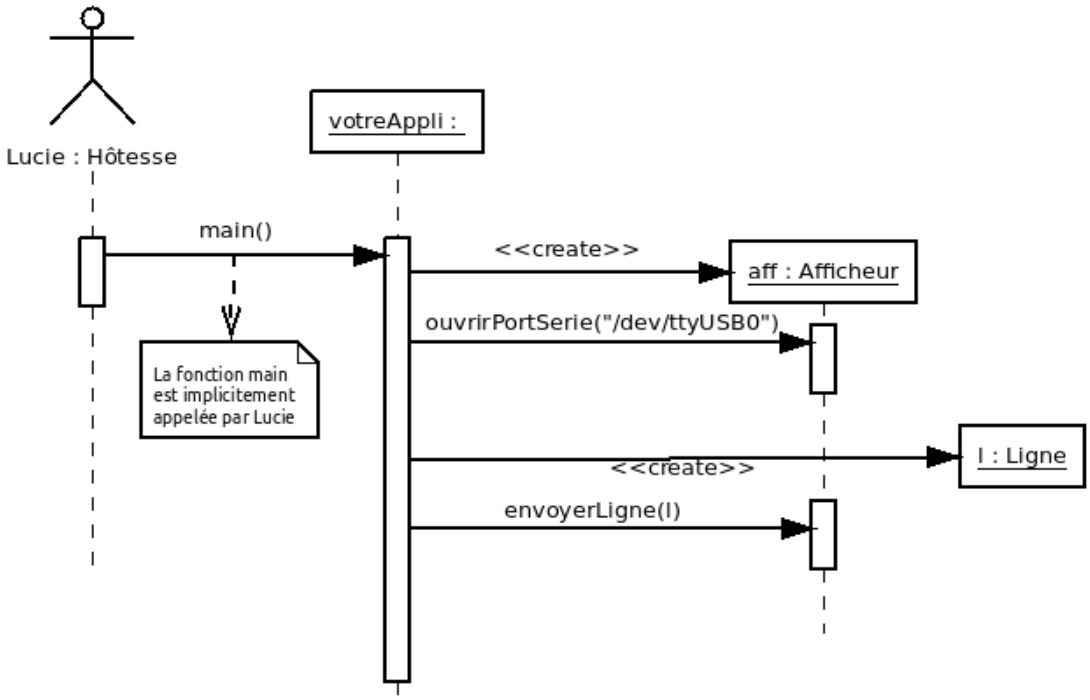
Dans cette activité, nous allons utiliser la classe **Ligne** qui va nous faciliter la mise en place d'un trame à envoyer à l'afficheur. Cette classe Ligne permet de gérer plus facilement le message à afficher et les effets à appliquer.

### Première utilisation de la classe Ligne : afficher un message par défaut...

Dans un premier temps, nous allons créer une ligne avec le constructeur sans argument Ligne() puis afficher cette ligne.

1. En utilisant la documentation html de la classe Ligne, expliquez ce que fait le constructeur sans argument de cette classe. En déduire la trame qui sera créée.

Le diagramme de séquence suivant décrit la création d'un objet de type Ligne et son envoi à l'afficheur :

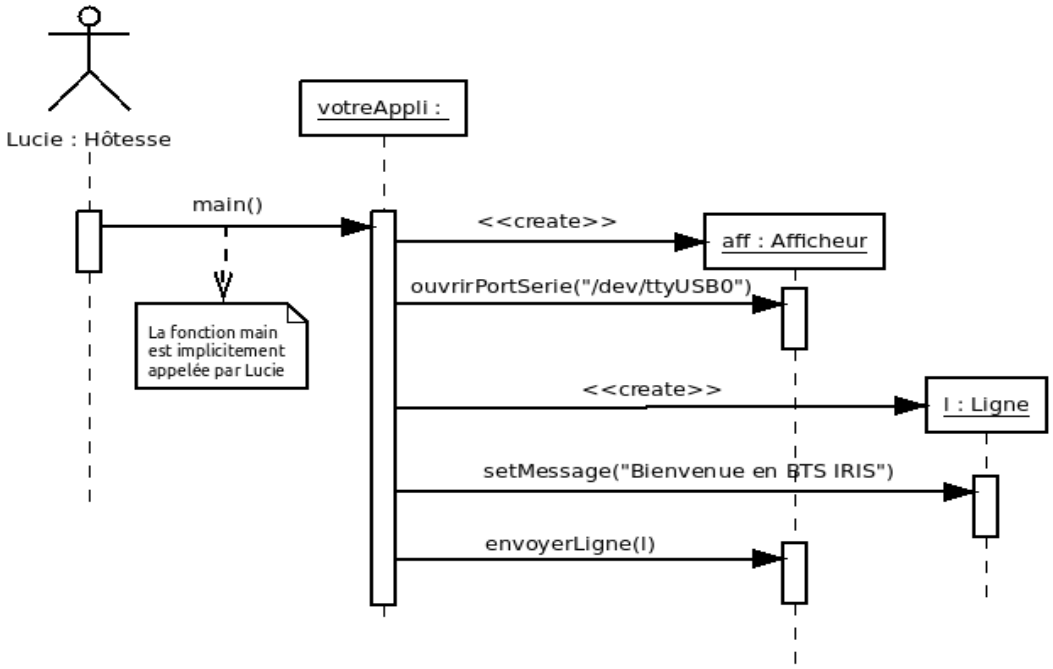



2. Modifiez votre programme pour qu'il ajoute la création d'un objet de type Ligne et qu'il envoie la ligne ainsi créée à l'afficheur.
3. Commentez chaque ligne de votre code.
4. Compilez et testez votre application.

### Personnaliser le message à afficher

Dans la classe Ligne, la méthode setMessage() permet de modifier le message qui sera affiché sur l'afficheur.

5. Utilisez la méthode setMessage() de la classe Ligne pour modifier le message à afficher, comme indiqué dans le diagramme de séquence ci-dessous :




6. Commentez chaque ligne de votre code.
7. Compilez et exécutez votre programme.
8. Ajoutez le code actuel de votre fichier main.cpp dans votre compte-rendu.

## Saisir manuellement le message à afficher

Nous allons maintenant permettre à l'utilisateur de saisir son propre message. Voici la procédure à suivre :

1. Dans le main, déclarez une chaîne de caractères (nommez-la monTexte par exemple) qui contiendra le texte que saisira l'utilisateur.
2. Invitez l'utilisateur à saisir son texte en utilisant cout.
3. Saisissez les caractères saisis dans la chaîne de caractères (dans monTexte) avec cin.
4. Passez la chaîne de caractères saisie (monTexte) à la méthode setMessage().

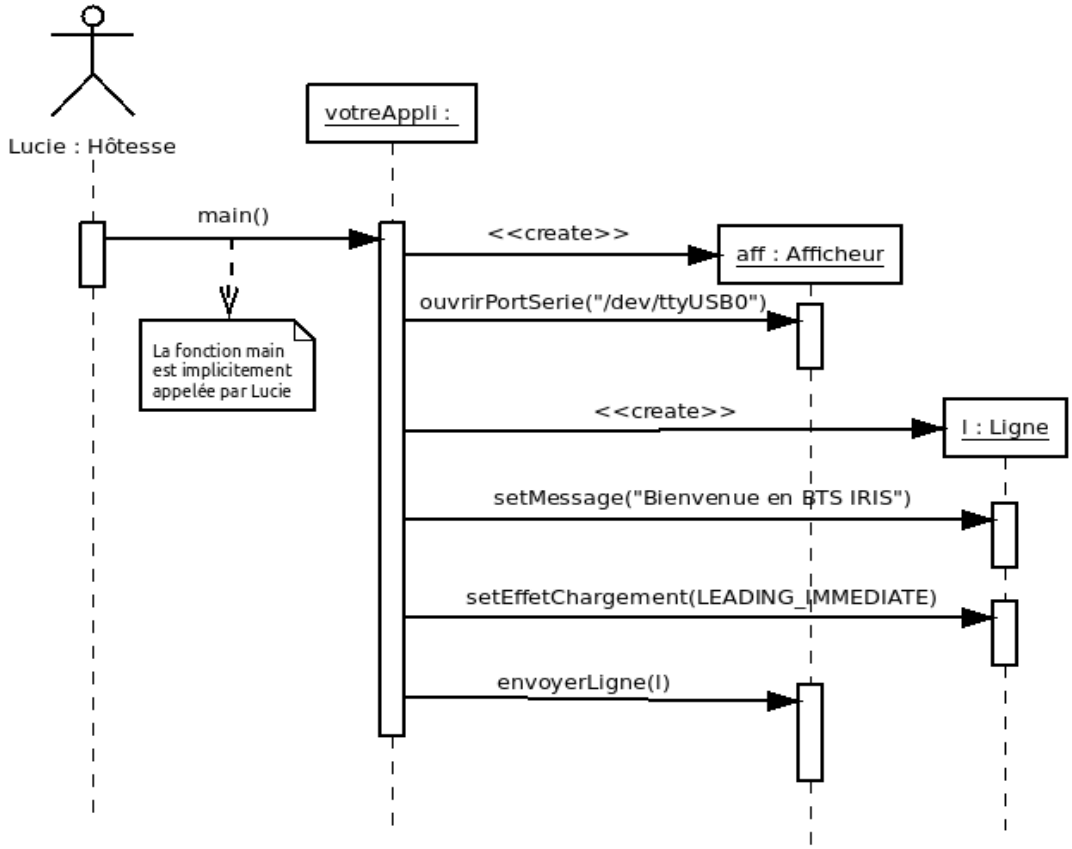
9. Modifiez votre code pour tenir compte de ces contraintes. N'oubliez pas les commentaires !
10. Compilez, exécutez et faîtes valider votre application par votre professeur.
11. Ajoutez le code de votre programme dans votre compte-rendu.

## Modifier l'effet de chargement

Dans cette partie, nous allons utiliser la méthode de la classe Ligne permettant de modifier l'effet de chargement ou d'apparition du message. Cette méthode est *setEffetChargement()*.

12. En utilisant la documentation html de la classe Ligne, indiquez le type de données attendu en argument de la méthode setEffetChargement().

Le diagramme de séquence suivant montre un exemple d'utilisation de cette méthode :




13. Modifiez votre code pour tenir compte de ces contraintes. N'oubliez pas les commentaires !
14. Compilez, exécutez et testez votre application. Vérifiez qu'elle fonctionne correctement.

## Pour terminer en beauté : afficher la température

On désire maintenant dans cette partie afficher la température actuelle. La température sera saisie par l'utilisateur de l'application. Le message ressemblera à : « Temp : 9,5 °C ». Les effets d'affichage seront les suivants :

- l'apparition du texte se fera de haut en bas ;
- le texte sera clignotant;
- le temps d'affichage sera de 10 secondes
- le texte disparaîtra de haut en bas.

15. Dessiner le diagramme de séquence de cette application.
16. Modifiez votre code pour tenir compte de ces contraintes. N'oubliez pas les commentaires !
17. Compilez, exécutez et testez votre application. Vérifiez qu'elle fonctionne correctement.

# Activité 5A – Une application graphique intuitive

## L'objectif

L'objectif de cette partie est de mettre en place un application graphique intuitive permettant un affichage d'informations sur l'afficheur à LED.

L'application disposera de deux modes de fonctionnement : un en réseau et l'autre en USB. Nous nous concentrerons dans un premier temps sur le fonctionnement en réseau. Ceci vous permettra à tous de développer sur vos machines une application graphique qui enverra par le réseau un message au poste d'afficheur, lui-même retransmettra ce message à l'afficheur en USB. Une fois l'application fonctionnelle en mode réseau, vous la testerez en USB.

## L'envoi d'une ligne en mode réseau

La classe Afficheur dispose de méthodes permettant de communiquer avec le serveur UDP fonctionnant sur le poste de l'afficheur. L'adresse IP et le port de ce serveur sont précisés dans la description du matériel dans les premières pages de ce document.

La méthode *setAdresseIPDuServeurUDP()* permet de modifier la valeur de l'adresse IP de ce serveur UDP. Par défaut, la valeur de l'adresse IP du serveur est 172.20.182.10. Voir la documentation de l'API pour plus de détail à ce sujet.

La méthode *setPortDuServeurUDP()* permet de modifier la valeur du port du serveur UDP. Par défaut, la valeur du port du serveur UDP est 4321. Voir la documentation de l'API pour plus de détail à ce sujet.

La méthode *envoyerLigneParUDP()* permet d'envoyer un objet de la classe Ligne au serveur UDP. Lorsque le serveur UDP reçoit cette Ligne, elle est retransmise à l'afficheur en USB. Voir la documentation de l'API pour plus de détail à ce sujet.

## Laisser parler votre imagination...

1. Dessiner une IHM la plus complète possible pour permettre à notre hôtesse d'accueil de saisir n'importe quel message, avec n'importe quel effet d'affichage (entrée, sortie, durée, etc.), n'importe quelle couleur et n'importe quel caractère.

## Un simple bouton

2. Créez un nouveau projet d'application graphique avec votre EDI.
3. Copiez dans votre dossier de projet les fichiers sources suivants : Afficheur.h, Afficheur.cpp, Ligne.h, Ligne.cpp, PortSerieAfficheur.h, PortSerieAfficheur.cpp, DateHeure.h et DateHeure.cpp. Ajoutez ensuite ces fichiers à votre projet.
4. Ajoutez ensuite dans votre IHM un simple bouton qui enverra une Ligne au serveur UDP.

Pour rendre votre IHM fonctionnelle, il faut ajoutez deux attributs dans votre classe gérant la boîte de dialogue.

5. Pour C++ Builder, dans le fichier Unit1.h, allez dans la zone de déclaration des attributs privés de la classe TForm1. Ajoutez la déclaration des 2 attributs suivant :

```
Afficheur aff ;
Ligne l ;
```

6. Ajoutez les fichiers d'inclusions nécessaires en haut du fichier Unit1.h.
7. Dans le constructeur de la classe TForm1, indiquez les paramètres par défaut du serveur UDP en appelant les méthodes *setAdresseIPDuServeurUDP()* et *setPortDuServeurUDP()* de l'objet aff.
8. Maintenant, lors du clic sur le bouton que vous venez de créer, vous allez envoyer une Ligne contenant votre prénom au serveur UDP. Il faudra appeler la méthode *setMessage()* de l'objet l et spécifier votre prénom. Puis vous appellerez la méthode *envoyerLigneParUDP()* de l'objet aff.

## Ajout progressif de fonctionnalité

Vous allez maintenant ajouter progressivement les fonctionnalités que vous avez dessiner à la

première question de cette activité. Ajoutez ces fonctionnalités une par une, et testez-les au fur et à mesure.

### Saisi du message

La première fonctionnalité à ajouter est la saisie du message à afficher.

9. Ajoutez un zone de texte. Limitez nombre de caractères pouvant être saisi en fonction de la capacité d'affichage de l'afficheur.
10. Modifiez votre code pour envoyer le texte saisi.
11. Testez et validez vos modifications.

### Saisi des effets

Vous allez maintenant ajouter la possibilité à l'hôtesse de modifier l'effet de chargement du message.

12. Ajoutez la possibilité de choisir un effet de chargement.
13. Modifiez votre code pour en tenir compte.
14. Testez et validez vos modifications.
15. Faîtes maintenant de même pour l'effet de sortie, l'effet d'affichage ainsi que la durée de l'affichage. Cherchez à rendre votre application la plus ergonomique possible. Testez et validez vos modifications.

### Gestion des couleurs

Nous voulons maintenant permettre à notre hôtesse de choisir la couleur du texte. Dans un premier temps, cette couleur s'appliquera à tout le texte.

16. Ajoutez la possibilité de choisir une couleur de texte.
17. Modifiez votre code pour en tenir compte.
18. Testez et valider vos modifications.
19. Bonus : trouvez une solution pour définir la couleur de chaque caractères du texte à afficher.

### Gestion des caractères spéciaux

Nous voulons enfin gérer l'affichage des caractères spéciaux. Si le message contient un 'é' par exemple, il devra s'afficher correctement.

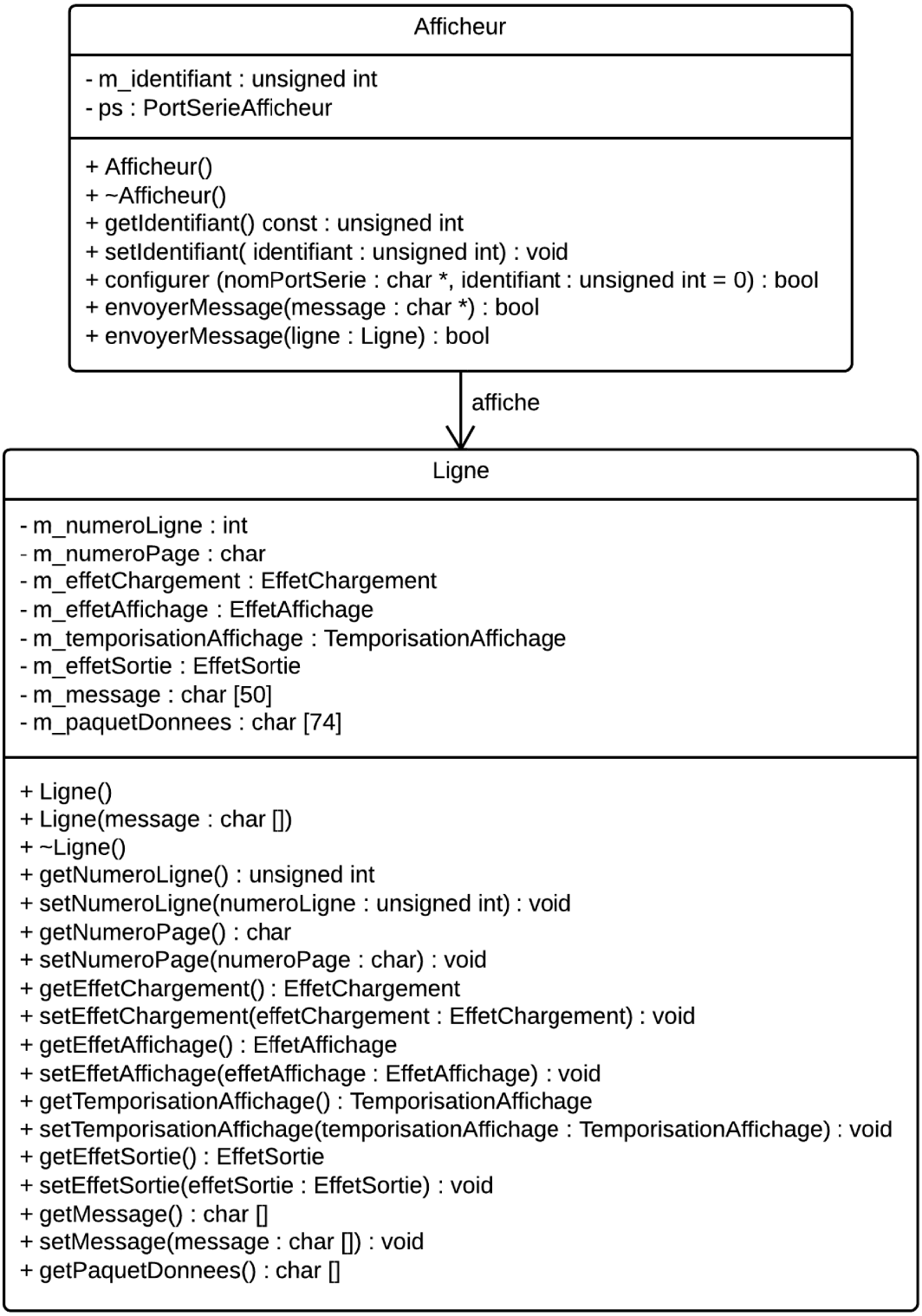
20. En analysant la documentation du constructeur de l'afficheur, indiquez quels caractères envoyés à l'afficheur à la place du 'é'.
21. Trouvez une solution pour remplacer automatiquement tous les caractères 'é'.
22. Testez et validez.
23. Faire de même pour les autres caractères spéciaux.

### Envoi par le réseau ou par USB

Proposez à l'hôtesse d'accueil de choisir entre un envoi par USB ou un envoi par le réseau. Dans la cas de l'USB, elle devra indiquer le port série à utiliser. Dans le cas du réseau, elle devra indiquer l'adresse IP et le port du serveur UDP.

24. Ajoutez les éléments dans l'IHM permettant à l'hôtesse de faire son choix.
25. En fonction de son choix, envoyer le message au serveur UDP ou directement par le port série.

# Annexe 1 - Diagramme des classes Afficheur et Ligne



# Annexe 2A

## Description des fichiers de l'application et Édition de main.cpp

L'objectif de cette annexe est de décrire les éléments constitutifs d'un projet en C++ sous linux, sans utiliser d'EDI (c'est-à-dire en utilisant simplement un éditeur de texte et un terminal).

Les fichiers sources de l'application se trouvent dans le répertoire sources.

C'est le poste afficheur (possédant un OS Linux) qui compilera et exécutera votre application.

### Copie des fichiers sources dans votre répertoire de travail

1. Pour travailler proprement, copiez le répertoire sources dans votre répertoire de travail.

### Description des fichiers

Vous ne partez pas de rien pour créer ce projet (je dirai même qu'on vous fourni tout, ouf...). Le répertoire nommé sources contient les fichiers suivants :

- Afficheur.h et Afficheur.cpp : les fichiers sources écrits en C++ décrivant la classe Afficheur.
- Ligne.h et Ligne.cpp : les fichiers sources écrits en C++ décrivant la classe Ligne.
- PortSerieAfficheur.h et PortSerieAfficheur.cpp : les fichiers sources en C++ décrivant la classe PortSerieAfficheur.
- main.cpp : fichier source écrit en C++ contenant la fonction *main()*.
- Makefile : un fichier décrivant les instructions de compilation du projet.

### Édition des fichiers sources

Pour éditer et modifier les fichiers sources, vous pouvez utiliser l'éditeur de texte GEdit. Vous en aurez notamment besoin pour éditer le fichier main.cpp que vous aurez à compléter par la suite.

2. Éditez le fichier main.cpp avec l'éditeur GEdit (pour Linux) ou Notepad++ (pourWindows). Modifier maintenant le contenu de ce fichier en suivant l'activité proposée.

# Annexe 3A

## Compilation et exécution sur le poste Afficheur

Cette annexe décrit 3 étapes à suivre pour compiler et exécuter votre programme sur le poste Afficheur :

1. Créer un répertoire à votre nom sur le poste Afficheur.
2. Transférer vos fichiers dans ce répertoire.
3. Compiler et exécuter votre programme.

Décrivons maintenant ces 3 étapes.

### Création d'un répertoire personnel sur le poste Afficheur

3. Ouvrez un terminal distant du poste Afficheur avec SSH (voir Annexe 4)
4. Déplacez-vous dans le répertoire Documents avec la commande :

```
cd Documents
```

5. Créez un répertoire à votre nom à cet endroit avec la commande mkdir :

```
mkdir VotreNom
```

6. Déplacez-vous maintenant dans ce répertoire :

```
cd VotreNom
```

Vous disposez maintenant d'un répertoire à votre nom dans le répertoire Documents du poste Afficheur. C'est dans ce répertoire que vous mettrez vos fichiers sources et que vous lancerez la compilation.

### Transfert des fichiers sur le poste Afficheur

Vous allez utilisez le protocole FTP pour transférer vos fichiers. Filezilla est un client FTP qui nous sera très utile.

7. Installez Filezilla s'il n'est pas installé sur votre machine.
8. Lancez Filezilla en précisant l'adresse IP du poste Afficheur, l'identifiant et le mot de passe (voir la section Matériel en première page de ce document)
9. Une fois connecté, déplacez-vous dans votre répertoire (Documents/VotreNom)
10. Faîtes un Drag & Drop de tous vos fichiers sources (*.h, *.cpp, Makefile) dans votre répertoire personnel du poste afficheur.

### Compilation et exécution

11. Lancez la compilation avec votre terminal distant (par SSH). Si vous vous trouvez dans votre répertoire de travail, tapez la commande :

```
make
```

12. S'il n'y a pas d'erreur de compilation, lancez l'exécution de votre programme avec la commande :

```
./afficheur
```

### Et après...

Par la suite, si vous modifiez un fichier source de votre projet, transférez uniquement ce fichier sur le poste afficheur, recompilez et exécutez...

# Annexe 3B – Compilation et exécution de l'application

L'objectif de cette annexe est de vous aider à compiler votre application et à la transférer sur le poste de l'afficheur à LED pour l'exécuter.

## Compilation de l'application

La compilation de votre application se fait à partir d'un terminal.

Ouvrez un terminal (Applications → Accessoires → Terminal).

Déplacez-vous dans votre répertoire de travail.

Par exemple, si votre répertoire de travail se nomme Prof/Afficheur

## Transfert et exécution de l'application sur le poste de l'afficheur à LED

Le poste de l'afficheur possède lui aussi un système d'exploitation Ubuntu. Il sera donc capable d'exécuter le programme que vous venez de créer. Mais pour cela, il faudra transférer l'exécutable que vous venez de

### Transfert de l'application

Nous pourrions très bien utiliser une clé usb pour transférer l'application d'un poste à un autre, puis l'exécuter. D'une part ce serait fastidieux de le faire à chaque fois que l'on modifie notre programme. De plus, si vous deviez tous le faire en même temps, il risque fort d'y avoir un gros embouteillage sur le poste de l'afficheur...

Puisque tous les postes sont sur le même réseau, nous allons utiliser une commande linux pour faire le transfert d'un fichier. Cette commande se nomme *scp*. Le *cp* signifie *copy* (copie) et le *s* signifie *secure* (sécurisée). Elle permet de copier un fichier d'un poste à un autre en toute sécurité. Cette commande utilise le protocole SSH dont nous reparlerons plus tard.

Voici un exemple de la copie du fichier local nommé *afficheurLED.pdf* se trouvant dans le répertoire Afficheur sur un poste dont l'adresse IP est 192.168.1.140 et dont l'utilisateur est *etudiant*.

```
scp Afficheur/afficheurLED.pdf etudiant@192.168.1.140:/home/etudiant/Afficheur/
```

Faîtes la même chose avec votre exécutable. Transférez-le sur le poste *192.168.1.244* avec le compte *etudiant* et dont le mot de passe est aussi *etudiant*.

### Exécution de l'application

Pour les mêmes raisons que précédemment, vous allez lancer l'exécution de votre programme sans bouger de votre chaise, en utilisant le réseau. Le protocole que vous allez utiliser se nomme SSH (Secure Shell). Ce protocole permet de d'ouvrir un terminal d'une machine à partir d'un poste distant.

Pour ouvrir un terminal d'une machine à distance, il faut connaître l'adresse IP de la machine, posséder un compte et un mot de passe. Vous connaissez tout ça pour notre poste afficheur !

La commande est tapée est alors celle-ci :

```
ssh etudiant@192.168.1.244
```

Saisissez ensuite le mot de passe du compte *etudiant*.

Vous vous trouvez maintenant dans un terminal du poste de l'afficheur.

Déplacez-vous dans le répertoire Afficheur où se trouve votre exécutable.

```
cd Afficheur
```

Lancez votre exécutable avec la commande suivante (en changeant MonProgramme par le nom du vôtre) :

```
./MonProgramme
```

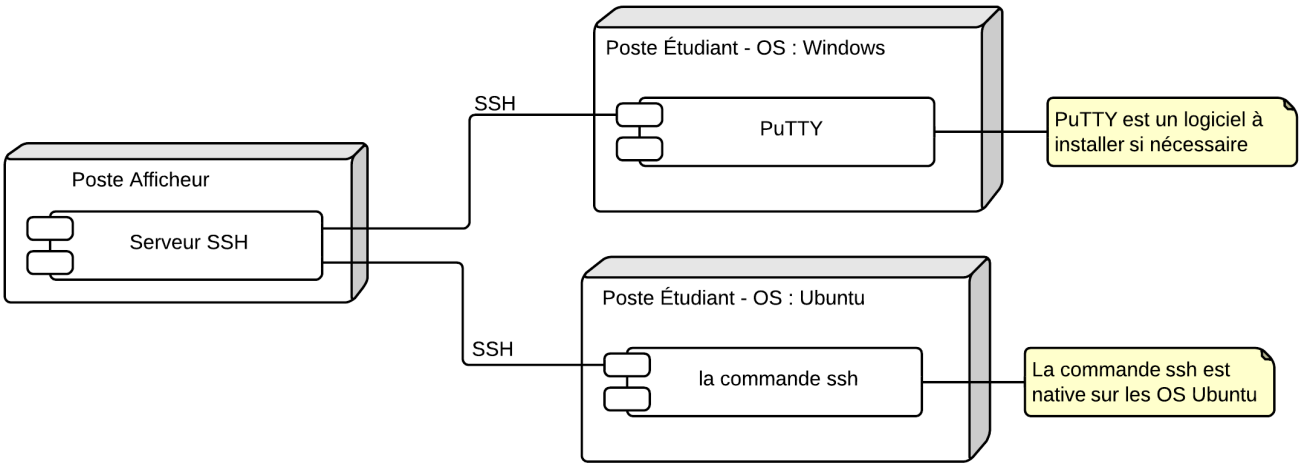
# Annexe 4 – Un terminal distant avec SSH

## Présentation

Le protocole SSH (*Secure Shell*) à de nombreuses applications. Nous parlerons ici que de la première d'entre elles : ouvrir un terminal (ou *Shell* ou interface en ligne de commande) sur un poste distant de façon sécurisée (*Secure*).

## Prérequis pour disposer un terminal distant

Le poste distant sur lequel on désire ouvrir un terminal doit posséder un serveur SSH qui écoute les demandes de connexions sur le port 22. Le poste client doit posséder une application cliente SSH : sous Linux nous disposons de la commande ssh ; sous Windows, nous utiliserons l'application PuTTY. Le diagramme de déploiement suivant explique l'architecture utilisée.




## Ouverture d'un terminal distant

Pour ouvrir un terminal distant, il faut disposer :

- l'adresse IP ou le nom de la machine distante ;
- posséder un compte utilisateur et un mot de passe sur cette machine.

### Sous Windows

Sous Windows, indiquez l'adresse IP et le numéro de port du serveur SSH (port 22). Un terminal s'ouvre si la connexion s'est établie. Saisir alors le nom d'utilisateur puis le mot de passe.

### Sous Linux

Sous Linux, tapez la commande :

```
ssh utilisateur@adresseIPPosteAfficheur
```

Saisir ensuite le mot de passe comme demandé.

Vous obtenez alors un terminal opérationnel dans lequel vous pouvez saisir vos commandes.