Université d'Orléans **1999-2000**
IUP'3 Informatique

Projet Système:
Mini-Web

1 Objectifs

L'objectif de ce projet est de développer un système d'information sur le modèle du web. Les principaux éléments sont le navigateur, le serveur et les bases de données:

- Le navigateur permet de récupérer des pages html chez un serveur et de les afficher. Dans le cas où la page affichée contient un formulaire que l'utilisateur peut remplir, le navigateur doit aussi être capable de transmettre ces données au serveur.
- Le serveur auquel se connecte un ou plusieurs navigateurs. Le serveur gère des pages html qu'il transmet au navigateur les ayant demandées. Le serveur est aussi capable de récupérer les données d'un formulaire remplis au niveau d'un navigateur et de déclencher l'exécution d'un programme traitant ces données, de récupérer le résultat (qui peut être une nouvelle page html créée dynamiquement) et de le transmettre au navigateur. Ces programmes et le serveur s'interfacent suivant le modèle CGI (Common Gateway Interface).
- Des bases de données auxquelles le serveur peut soumettre des requêtes par l'intermédiaire des programmes traités suivant le modèle CGI (appelés programmes CGI). Les réponses aux requêtes sont récupérées par le programme CGI qui génère alors une page html qu'il transmet au serveur.

2 Spécification

Une spécification détaillée des applications à développer est donnée dans la suite. Cette spécification est organisée en plusieurs étapes qu'il vous est conseiller de suivre dans l'ordre, vous permettant ainsi d'aborder les différents problèmes de manière progressive.

2.1 Le protocole mini-http

Un élément important du système d'information est le protocole utilisé pour permettre au navigateur et au serveur de dialoguer. Vous devrez donc définir un tel protocole que nous appelons mini-http. Vous pouvez pour cela vous inspirer du protocole utilisé pour le web: http (voir `www.w3c.org`).

2.2 Le navigateur

Vous implanterez un navigateur capable de se connecter à une adresse de serveur donnée, de récupérer une page html et de l'afficher en tenant compte des instructions html de formatage qu'il contient. Le navigateur doit aussi être capable d'afficher une page html stockée localement sur votre machine. Une page html est un fichier texte contenant des tags html. Un tag html permet de délimiter une zone de texte pour laquelle une formatage spécifique est demandé. Les tags ont la syntaxe suivante: `<TAG>` pour le début et `</TAG>` pour la fin. Certains tags peuvent avoir des attributs auxquels des valeurs sont associées. Un tag à deux attributs a la syntaxe suivante: `<TAG attribut1="valeur1" attribut2="valeur2">` pour le début et `</TAG>` pour la fin. Pour certains tags la marque de fin n'est pas nécessaire. Les tags que doit reconnaître le navigateur sont les suivants:

- `HTML`: Début et fin d'un fichier HTML.
- `HEAD`: En-tête du fichier. Les informations contenues dans cette zone de texte ne sont pas affichées par le navigateur.
- `BODY`: Corps du fichier.
- `P`: Paragraphe de texte.
- `B`: Texte en gras.
- `A`: Ce tag permet de définir un lien vers une page html spécifiée par l'attribut `href`. La syntaxe est: `<A href="url">lien</A>`. Le texte `lien` est mis en relief par rapport à du texte normal (classiquement avec la couleur bleue). Lorsque l'utilisateur clique sur `lien` alors le navigateur utilise l'information `url` pour identifier le serveur auquel il doit se connecter et la page html qu'il doit demander à ce serveur. Le format adopté pour l'url dépendra du protocole mini-http que vous choisirez. Par exemple sur le web, la page `index.html` du serveur de l'université d'Orléans peut être désignée par l'url `http://www.univ-orleans.fr:80/index.html`.

Les tags non reconnus par le navigateur sont ignorés. Pour plus de détails sur html se référer au site `www.w3c.org`.

Le navigateur pourra en grande partie être développé et testé en utilisant simplement sa capacité à afficher des pages locales.

2.3 Le serveur

Vous implanterez un serveur capable de communiquer avec plusieurs clients (typiquement des navigateurs) en parallèle (pseudo-parallélisme pour la plupart des machines). Les communications entre processus seront gérées sur le modèle client/serveur par l'intermédiaire des sockets. Le serveur a un socket "passif" qui attend des demandes de connexions. Le client possède un socket qui demande une connexion au serveur. Dès que la demande de connexion parvient au socket passif, ce socket instancie un socket "actif" connecté au socket client. Le dialogue de demandes de pages html et envois de pages html a alors lieu par l'intermédiaire du socket client et du socket actif du serveur. Pour que le serveur soit capable de communiquer avec plusieurs clients en parallèle chaque socket actif du serveur doit être associé à un processus ou thread différent. Ainsi chaque fois qu'une demande de connexion parvient au socket passif du serveur, un thread ou processus est créé pour prendre en charge le socket actif instancié.

La commande `telnet` permet de se connecter à un serveur utilisant des sockets. Ainsi vous pouvez développer et tester votre serveur indépendamment du navigateur en établissant une connexion via `telnet`. Par exemple pour se connecter au serveur de l'université d'Orléans il est possible d'utiliser la commande `telnet www.univ-orleans.fr 80` (80 correspond au numéro du port d'écoute du socket passif). Puis an entrant la commande GET vous recevrez la page `index.html` du site.

Vous trouverez des informations sur les sockets avec :

- Le man pour la programmation des sockets en C ou C++ :
  `man tcp`, `man -s3N socket`.
- Le tutorial Java:
  `http://java.sun.com/docs/books/tutorial/index.html`
- Le site `www.univ-orleans.fr/SCIENCES/LIFO/Members/virot/enseignement/`

2.4 Les formes et les programmes CGI

Dans une seconde étape vous améliorerez mini-http, le serveur et le navigateur pour pouvoir exécuter des programmes CGI.

2.4.1 Le protocole mini-http

Mini-http doit maintenant intégrer le fait que le navigateur peut émettre des demandes de pages, mais aussi envoyer des messages demandant l'exécution de programmes CGI. Mini-http doit donc spécifier le format de ces messages.

2.4.2 Le navigateur

Le navigateur doit reconnaître les tags dits de forme suivants qui vont permettre d'afficher des formulaires que l'utilisateur peut remplir et envoyer au serveur:

- FORM: Délimite une forme. L'attribut `action` définit le programme CGI que doit exécuter le serveur lorsqu'il aura reçue le message correspondant à cette forme.
- INPUT: Le tag INPUT n'a pas de forme de fin. Les attributs qui lui sont associés définissent son rôle:
  - Si l'attribut `type` a la valeur:
    - `text` alors le tag INPUT correspond à une zone où l'utilisateur peut saisir du texte.
    - `reset` alors le tag INPUT correspond à un bouton de ré-initialisation de la forme.
    - `submit` alors le tag INPUT correspond à un bouton. Lorsque l'utilisateur clique sur ce bouton, le message basé sur les données de la forme est construit et transmit au serveur.
  - L'attribut `name` permet d'attribuer des noms aux différents tags INPUT. Ces noms sont utilisés pour construire le message envoyé au serveur
  - L'attribut `value` définit:
    - Le nom affiché sur le bouton correspondant.
    - Le texte affiché initialement ou après ré-initialisation dans la zone de texte correspondante.

Pour une spécification détaillée des formes en html se référer au site `www.w3c.org`.

2.4.3 Le Serveur

Le serveur doit être capable de reconnaître les demandes d'exécutions de programmes CGI, d'identifier les informations à transmettre au programme CGI et de récupérer la réponse du programme. Pour le web, il existe deux méthodes pour communiquer avec un programme CGI: GET et POST. Par la méthode POST le serveur va transmettre au programme CGI une chaîne de caractères du type `nom_input1=valeur_input1?nom_input2=valeur_input2` si la forme a deux tags INPUT (en plus des boutons "submit" et "reset"). Le programme CGI lit cette chaîne sur son entrée standard et utilise cette information pour définir sa réponse qu'il écrit sur la sortie standard. La réponse émise par le programme CGI est récupérée par le serveur qui la retransmet au navigateur. Dans la plupart des cas cette réponse correspond à une page html. Noter qu'un programme CGI doit pouvoir être exécuté par le serveur quelque soit sont langage de programmation (JAVA, C++, Perl, shell-script...).

Vous trouverez de la documentation concernant CGI sur:

- `www.univ-orleans.fr/SCIENCES/LIFO/Members/virot/enseignement/`
- `www.cgi-resources.com/`
- `http://www.cgi101.com/class/`

2.4.4 Exemple

Voici un exemple de page html contenant une forme avec un bouton "submit", "reset" et une zone de texte contenant initialement le texte "Bonjour":

```
<HTML>
<BODY>
<P> <B>Une forme</B></P>

<P>Cette forme permet de saisir un texte, de le transmettre au
 serveur qui déclenche  l'exécution de programme.cgi:</P>

<FORM action="programme.cgi" method="post">
<INPUT type="text" name="texte" value="Bonjour">
<INPUT type="submit" value="Envoyer">
<INPUT type="reset" value="Effacer">
</FORM>
</BODY>
</HTML>
```

Si l'utilisateur clique sur le bouton affichant "Envoyer" alors le serveur va demander l'exécution du programme CGI "programme.cgi" et lui transmet la chaîne de caractères "texte=Bonjour". Le programme CGI lit sur son entrée standard cette chaîne et crée dynamiquement une nouvelle page html contenant le message reçu (dans la limite de 20 caractères) qu'il écrit sur sa sortie standard. Le serveur fait alors suivre la page vers le navigateur du client. Voici ce programme CGI écrit en C:

```
#include <stdio.h>
```

```
int  main()
{
    char texte[20]="                    ";

    printf("Content-Type:text/html\n\n");
    printf("<HTML><BODY>");
    scanf("texte=%s",texte);
    printf("<P>Le texte  envoyé est: %s</P>",texte);
    printf("</BODY></HTML>");
}
```

Remarquer que ces exemples se conforment au langage html et au protocole http standard (voir l'attribut `method="post"` ou l'en-tête `"Content-Type:text/html"`). Votre navigateur et serveur doivent pouvoir exécuter ces deux programmes une fois adaptés au langage html et au protocole mini-http adoptés.

2.5 Les bases de données

Le projet doit également intégrer un module pour gérer l'accès et la mise à jour d'une base de données. L'objectif de l'intégration de ce module est de pouvoir envoyer des requêtes, à partir une page html, vers une base de données de votre choix (Mysql, Oracle ou Access).

Pour cela, la page html contient une forme qui va permettre de transmettre la requête au serveur. Le serveur déclenche alors l'exécution d'un programme CGI. Ce programme va consulter et éventuellement mettre à jour les tables de la base de données et transmettre une réponse vers le navigateur (via le serveur). La base de données est formée essentiellement des tables suivantes :

- une table de compagnies aériennes (Nom de la compagnie, Numéro du vol, Date du vol, Nombre de places, Ville de départ, Ville d'arrivée, etc.),
- une table des hôtels (Nom de l'hôtel, Ville, Nombre de chambres, etc.),
- une table des agences de location de voitures (Nom de l'agence, Ville, Nombre de voitures, etc.),
- une table des réservations des billets d'avions (Numéro du client, Date de réservation, Numéro du vol, etc.),
- une table des réservations des voitures (Numéro du client, Date de réservation, Numéro de la voiture, Nombre de jours, etc.),
- une table des réservations d'hôtels (Numéro du client, Date de réservation, Numéro de chambres, Nombre de jours, etc.).

Pour simplifier on ne s'occupe que des réservations

- d'une seule compagnie aérienne n'ayant qu'un vol par jour *Paris / Rome*,
- d'une agence de location de voitures située à l'aéroport de Rome n'ayant qu'un type de voitures.
- dans un hôtel basé également de Rome avec un seul type de chambres.

Une documentations complète sur l'utilisation du systèmes de gestion de bases de données `Mysql` est disponible sur la page web : *http://www.minet.net/devel/mysql/*

L'avantage de ce système est qu'il est facilement interfacable avec différents programmes (C, Java, perl, CGI, etc.). Un compte pour chaque utilisateur sera créé pour accéder à l'un des serveurs de bases de données Mysql ou Oracles. L'utilisation d'Access ne nécessite pas de compte particulier.

Vous écrirez donc une page html et le programme CGI correspondant pour permettre la réservation, éventuellement combinée, de billets d'avion, de chambres d'hôtels et de voitures.

Votre serveur doit pouvoir traiter plusieurs connexions en parallèle. Il est donc possible que plusieurs programmes CGI soient exécutés en parallèle et donc que des requêtes concurrentes soient émises. Ces requêtes peuvent être adressées à différentes bases de données ou à la même base de données. Vous devez donc être `très attentif` aux problèmes de concurrence à ce niveau. En particulier, veillez à ce que les réservations soient correctement faites lors d'une demande combinée d'un billet d'avion, d'une voiture et d'une chambre pour une période donnée. Une telle requête ne doit être satisfaite que si les trois réservations sont possibles.

3 Réalisation du projet

Votre travail consiste à définir le protocole mini-http, à programmer un navigateur, un serveur capable d'exécuter des programmes CGI, un ou des programmes CGI pour la réservation combinée avion/voiture/hôtel. Votre travail devra être aussi professionnel que possible. Vous devez ainsi penser à la facilité de maintenance et d'extension de vos programmes.

- **Groupes.** Ce projet doit être fait par groupe de trois personnes (et pas plus).
- **Machines cibles.** Les langages de programmation et les librairies que vous utiliserez doivent conduire à des **programmes compilables et exécutables sur les machines enseignement**. Vous pouvez donc opter pour Java, C, C++, Java, ADA, etc, mais en faisant attention d'utiliser les mêmes versions que celles installées sur les machines enseignement.
- **Interopérabilité.** Vous devrez vous assurer de l'interopérabilité de l'ensemble des projets de la miage, c'est à dire que n'importe quel navigateur doit pouvoir se connecter et communiquer avec n'importe quel serveur. Pour cela les spécifications doivent être publiques et conduire à la rédaction d'un document collectif "Mini-WEB: Le Standard". Celui-ci contiendra notamment:
  - la spécification du protocole mini-http adopté
  - le langage html minimum implanté
  - toute autre information essentielle pour assurer l'intéropérabilité

  Il vous est conseillé de tenir à jour l'historique des versions successives.
- **Délais.** Vous devrez me rendre :
  1. Une version papier du document "Mini-WEB: Le Standard" rédigé avec un traitement de texte (une seul version pour l'ensemble de la licence et pas une par projet).

2. Une version papier de votre rapport de projet. Votre rapport, rédigé avec un traitement de texte, devra contenir:
   - Un mode d'emploi complet (installation, compilation, exécution et utilisation).
   - Une description précise des fonctionnalités implantées (ce que fait le projet).
   - Une description synthétique des algorithmes, en particulier de la gestion concurrente des bases de données.
   - Une partie où vous justifierez les grands axes de votre approche et où vous ferez état de la répartition des tâches dans le projet, des difficultés rencontrées, de la communications intra et inter projets, des extensions et améliorations envisageables, des problèmes dans l'élaboration des standards , etc.

   Soyez professionnels. Essayez de penser à l'équipe qui utilisera ce document pour maintenir ou améliorer votre projet. Adaptez votre présentation au public visé (utilisateur non débutant pour le mode d'emploi, programmeur pour les algorithmes,...)
3. Une copie électronique de vos programmes **clairement commentés** organisés comme suit:
   Vous sauvegarderez vos programmes dans un répertoire `Miage-WEB-NOM1-NOM2-NOM3` qui contiendra:
   - Un fichier README traitant de l'installation, la compilation et l'exécution des programmes.
   - Le répertoire `Navigateur` qui contiendra les programmes du navigateur.
   - Le répertoire `Serveur` qui contiendra les programmes du serveur.
   - Le répertoire `HTML` qui contiendra quelques exemples de pages html et une page html pour la réservation avion/voiture/hôtel.
   - Le répertoire `CGI` qui contiendra quelques exemples de programmes CGI et le(s) programme(s) CGI de réservation avion/voiture/hôtel.

- **Soutenances:**
  - Les projets seront à rendre avant la date précisée à l'adresse:
    `http://www.univ-orleans.fr/SCIENCES/LIFO/Members/bamha/Miage/`
    Quelques programmes utiles, pour la réalisation du projet, sont également disponibles sur la même page web.
  - La soutenance comprendra une présentation des différentes parties programmées ainsi qu'une démonstration permettant de juger des points forts de votre réalisation. Elle devra en outre faire ressortir votre participation personnelle à l'ensemble. Partagez le temps de parole...
- `www.univ-orleans.fr/SCIENCES/LIFO/Members/bamha/Miage/`
  donnera accès à ce texte et à toute autre information qui pourra s'avérer utile.

Vous pourriez avoir la tentation de récupérer sur le web des programmes équivalents à ceux demandés. Il est tout à fait acceptable que vous consultiez ce type de programmes, mais copier entièrement ou partiellement ces programmes ne sera en aucun cas toléré. De même, les échanges d'idées et de réflexions inter projets sont vivement souhaitées. Par contre chaque projet devra écrire ses propres programmes. Ne pas vous conformer à ces règles entraînera une baisse très importante de votre note.

Bon Travail