Examen de l'unité d’enseignement

Technologie pour les applications client-serveur

RSX 102

Durée 3 heures

Première session le 22 juin 2009

TOUS DOCUMENTS PAPIERS AUTORISES
TOUS SYSTEMES ELECTRONIQUES INTERDITS
(ordinateurs, organiseurs, téléphones, consoles, …)

sauf CALCULETTES

_____

Pour chaque question il est demandé une justification précise de votre réponse.
Le barème de ce problème correspond à une notation sur 20

Problème Programmation d'applications client-serveur pour un jeu multi-joueur en ligne

Vous intervenez dans une équipe qui développe un jeu multi-joueur en ligne de type jeu de quête à la première personne. Dans le jeu, le personnage à travers un périple dans un univers de jeu type moyen âge doit réaliser des épreuves, collecter des objets, interagir avec d'autres joueurs ou des avatars mais il n'y a pas de combats.

L'état du jeu est maintenu sur un serveur central. La mécanique de jeu est répartie entre la partie serveur, et une partie cliente pour chaque joueur. Le réseau utilisé est Internet, et les joueurs utilisent un terminal de type PC. Le PC exécute essentiellement la logique de jeu relative à l'interface.

Question 1 : Dans ce contexte, pour les interactions on se pose la question du protocole de transport à utiliser. La collecte d'objets étant essentielle à la bonne marche du jeu, une attention particulière est portée au protocole de transport. Les deux solutions usuelles sont TCP et UDP, quels sont les avantages de chacun des protocoles, on rappelle qu'il n'y a pas de contrainte temps réel **(1 point)**.

REPONSE

TCP offre la fiabilité (détection d'erreur et reprise sur erreur, remise en séquence, contrôle de flux, essentiellement) c'est un aspect clef, en effet, un joueur qui acquiert quelque chose dans le jeu ne peut le perdre que s'il ne se conforme pas à la règle du jeu, mais pas parce que l'implantation du jeu n'est pas fiable.

UDP est non fiable, donc, un joueur risque d'acquérir un objet mais de le perdre par la faute d'erreurs lors de la transmission. Le jeu aurait été un jeu de tir à la première personne comme Quake, ou CounterStrike par exemple, on aurait pu choisir UDP. En effet, le joueur tire des munitions à la suite, ce n'est pas nécessairement grave si une munition n'arrive pas au but, on peut toujours invoquer l'imprécision du tir, ou la défaillance d'une munition (ramassée par terre)… le même argument ne tient plus avec un tireur d'élite.

Non attendu dans la réponse : *le protocole TCP offre le contrôle de congestion qui permet de réguler la charge du réseau Internet, comme le jeu est en ligne (donc à travers tout internet), il est préférable d'utiliser TCP.*

Question 2 : On examine quelle est l'API (Application Programming Interface) la plus adaptée, dans un premier temps on s'intéresse à l'API socket. On suppose qu'on utilise le protocole TCP.

2.1 Quel est l'enchaînement de primitives de l'API socket qui doit être programmé en langage C côté serveur ? Vous donnerez votre réponse sous la forme d'un graphe. **(1 point)**

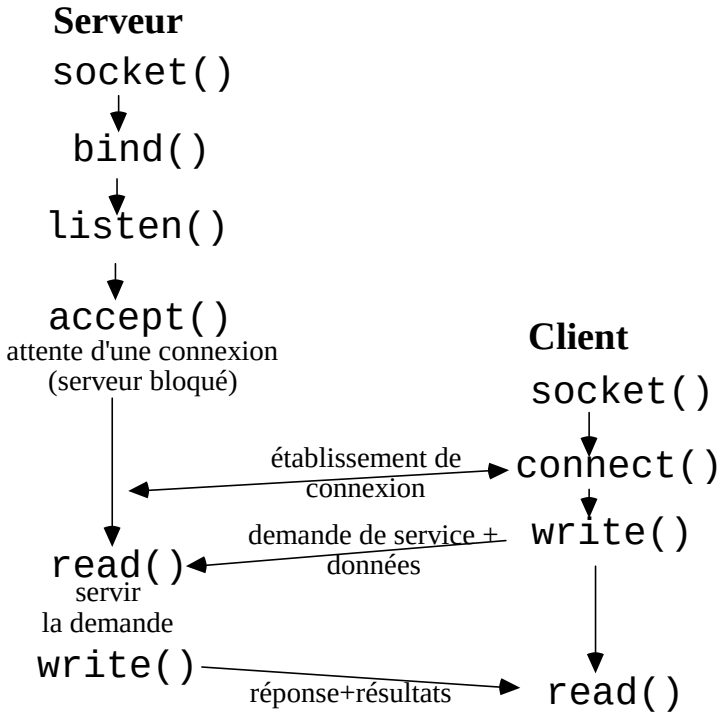
REPONSE

```
  Côté Serveur on enchaîne :
  Socket () /* pour créer une boite aux lettres */
  Listen () /* pour dimensionner la taille de la file d'attente
des demandes entrantes pour une ouverture de Connexion */
  Accept () /* qui met l'extrêmité TCP côté serveur en mode
ouverture de connexion passive */
  Lors des échanges on fera des send(), recv (), read (), write
()
  La fermeture de la connexion s'effectue par un Close (), on
peut utiliser un Shutdown () aussi.
```

2.2 Quel est l'enchaînement de primitives de l'API socket qui doit être programmé en langage C côté client ? Vous donnerez votre réponse sous la forme d'un graphe. **(1 point)**

```
  Côté Client on enchaîne :
  Socket () /* pour créer une boite aux lettres */
  Connect () /* pour faire une ouverture de connexion active */
  Lors des échanges on fera des send(), recv (), read (), write
()
  La fermeture de la connexion s'effectue par un Close (),on
peut utiliser un Shutdown () aussi.
```

```
  Pour résumer les deux automates, certains l'ont peu-être fait
sur la même figure, c'est bon !!! c'est plus judicieux.
```



2.3 Quelle primitive instancie la connexion TCP effectivement dans le réseau, et à quel moment précis est-elle utilisable par les programmes client et serveur ? **(1 point)**

REPONSE

C'est le connect() du client qui active la connexion TCP et permet de l'instancier effectivement. La connexion est établie au sens du protocole TCP quand le client sort du connect() et le serveur de l'accept().

2.4 Deux modèles de programmation sont possibles pour les échanges entre client et serveur:

- on crée une connexion, on effectue un transfert d'information, on ferme la connexion, c'est le modèle d'accès à une page web,
- on crée une connexion, elle est utilisable tout le temps de la partie de jeu entre le client et le serveur.

Donnez brièvement les avantages et les inconvénients de ces deux modèles de programmation**. (1 point)**

REPONSE

Dans le premier cas, les ressources monopolisées par la connexion ne sont pas utilisées longtemps. Par contre, il faut tout recréer à chaque interaction. Ce modèle est valable si les interactions ne provoquent pas des transferts de longue durée, et ne sont pas très fréquentes.

Dans le deuxième cas, on monopolise les ressources de la connexion pendant une longue durée, toute la partie. Ici, c'est le contraire de la première situation, chaque interaction peut être mise en œuvre plus rapidement car toutes les ressources sont déjà allouées.

Dans un jeu multi-joueurs en ligne, il va falloir dimensionner les serveurs en fonction de la fréquence des interactions estimées, du volume des données échangées … en fait, le choix de la solution dépend complètement du jeu.

2.5 Avec l'API socket, on se propose de transférer les données au format ASN.1. L'un des messages échangé concerne la description d'un autre personnage lors d'une rencontre. Le contenu de ce message se définit par les informations suivantes :

nom qui se décompose en : surnom, nom de sa tribu (chaînes de caractères)
identificateur d'un personnage dans le jeu (entier)
caractéristiques :
    royaume dont le personnage est issu (chaîne de caractères)
    suzerain (chaîne de caractères)
    pouvoir dont il dispose (chaîne de caractères)

La spécification ASN.1 de ce message est la suivante :

```
Personnage ::= [APPLICATION 0] IMPLICIT SEQUENCE
      {Nom,
      identificateur    [0] IMPLICIT INTEGER,
      caracteristiques[1]     SEQUENCE {
            royaume           [0] IMPLICIT VisibleString,
            suzerain          [1] IMPLICIT VisibleString,
            typedepouvoir     [2] IMPLICIT VisibleString}

Nom ::= [APPLICATION 1] IMPLICIT SEQUENCE {
      surnom      VisibleString,
      tribu       VisibleString}
```

Un personnage rencontré par le joueur est spécifié de la façon suivante :
{{lou, elfe},0016, {mforet, elroon, mage}} ici l'entier est en décimal

Le résultat correspondant aux données ci-dessus, généré en syntaxe de transfert suivant la codage BER (Basic Encoding Rules), est le suivant :

60	*				**personnage**
	61	0B			**nom**
		1A	03	6C6F75	**surnom** (lou)
		1A	04	656C6665	**tribu** (elfe)
	80	02	0010		**identificateur** (0016)
	A1	*			**caracteristiques**
		80	06	6D666F726574	**royaume** (mforet)
		81	*	656C726F6F6E	**suzerain** (elroon)
		82	04	6D616765	**typedepouvoir** (mage)

Les "*" correspondent à des longueurs de champs ou de sous-champs, comme le prévoit ASN.1. On vous demande de remplacer les "*" par les bonnes valeur en HEXADECIMAL. Quelle est la longueur totale des données envoyées ? **(2 points)**

REPONSE

60	***29***				**personnage (41)**
	61	0B			**nom**
		1A	03	6C6F75	**surnom** (lou)
		1A	04	656C6665	**tribu** (elfe)
	80	02	0010		**identificateur** (0016)
	A1	***16***			**caracteristiques (22)**
		80	06	6D666F726574	**royaume** (mforet)
		81	***06***	656C726F6F6E	**suzerain** (elroon)**(6)**
		82	04	6D616765	**typedepouvoir** (mage)

Question 3 : Toujours dans ce souci de choisir une bonne API on examine le paradigme de l'Appel de Procédure à Distance (APD, ou RPC pour Remote Procedure Call en anglais) sous différents formats possibles.

3.1 Si on choisit le protocole UDP, et que le serveur était programmé de telle manière qu'il soit sans état, quelle serait la contrainte sur la sémantique du RPC à mettre en œuvre. Expliquez brièvement pourquoi ? **(1 point)**

REPONSE

Trois sémantiques sont possibles dans un RPC :
"au moins une fois"*, le client invoque le serveur tant qu'il n'a pas de réponse, l'inconvénient est que plusieurs*

exécutions sur le serveur sont possibles … on a donc une contrainte d'idempotence des exécutions

***"exactement une fois"**, c'est une exécution transactionnelle, il faut garantir que l'exécution ne se produit qu'une et une seule fois, c'est coûteux*

***"au plus une fois"**, le client invoque une serveur, s'il a une réponse il sait que sa demande a été exécutée une fois pas plus, s'il n'a pas de réponse, sa demande a été exécutée une fois ou pas du tout*

avec UDP, on ne peut mettre en œuvre que la sémantique "au moins une fois" sinon, on offre une sémantique "peut-être"

3.2 Si on choisit le protocole TCP qu'on met en œuvre le mécanisme d' "IDLE MESSAGE" (option KEEPALIVE, qui veut dire que périodiquement des messages vides sont envoyés sur la connexion TCP), et que le serveur est avec état, quelle sémantique de RPC peut-on obtenir? Expliquez brièvement pourquoi ? **(1 point)**

REPONSE

Cf ci-dessus, avec TCP qui est fiable et l'option KEEPALIVE on sait si la requête est arrivée ou non, et on sait si la connexion est rompue ou non. On peut raisonnablement penser mettre en œuvre facilement une sémantique "au plus une fois" comme dans CORBA par exemple. On peut même envisager une sémantique "exactement une fois" si on rajoute un protocole de journalisation et de transaction à deux phases.

3.3 En choisissant d'implanter le jeu dans un navigateur et en utilisant une méthode permettant de mettre en ligne les informations portant sur un personnage à l'adresse URL http://jeu.cnam.fr/personnage.html, quel est le format des messages échangés via le protocole HTTP en utilisant la méthode GET pour obtenir une page html suivante **(2 points)**:

Personnage

- Nom :
  - surnom : lou
  - tribu : elfe
- identificateur : 0016
- Caracteristique
  - royaume : mforet
  - suzerain : elroon
  - type de pouvoir: mage

REPONSE

```
GET /personnage.html HTTP/1.0
Host: jeu.cnam.fr
```

Remarque : seul le champs d'entête host est obligatoire, d'autres champs peuvent toutefois être ajoutés

Réponse renvoyée par le serveur au client :

```
HTTP/1.0 200 OK
```

```
  date:  Mon, 22 Jun 2009 02:00:01 GMT
  Server: Apache
  Content-type: text/html
  Content-lenght: ...

  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
  <HTML>
  <HEAD></HEAD>
<H1> Personnage </H1>
<UL>
<LI> <I> surnom</LI>: lou </LI>
<LI> <I> tribu</LI>: elfe </LI>
<LI> <I> identificateur</LI>: 0016</LI>
<LI> <I> royaume</LI>: mforet </LI>
<LI> <I> suzerain</LI>: elroon</LI>
<LI> type de pouvoir: mage</LI>
</UL>
  <BODY>
  </BODY>
  </HTML>
```

Remarque : dans la réponse renvoyée par le serveur, le "status" de la requête 200 indique le succès de la requête du client qui est désormais acomplie;

3.4 On se pose la question d'écrire les interactions en utilisant WSDL, que donne la spécification correspondante pour une méthode *GET (personnage)* **(2 points)**

REPONSE

L'interaction utilisant les web services va nécessairement passer par un message http. La première partie de ce message sera celle d'une requête http standard de la forme :

Ligne de résumé de la requête

Plusieurs lignes dites d'en-tête de la forme variable: valeur

Un saut de ligne

La partie XML de la requête pour le web services. Cette partie est le corps du message (cf. HTTP).

***Consigne** : Compte tenu du cours, si les étudiants répondent tout cela on peut mettre les 2 points de la question.*

Evidemment on peut compléter par

Puisque la requête est une requête GET, la ligne de résumé de la requête sera de la forme :

GET arborescence menant au service web HTTP/1.0

Les lignes suivantes auront les caractéristiques d'une requête HTTP (User-Agent: …) mais ont sûrement des valeurs qui indiquent que le client est prêt à recevoir des réponses SOAP

comme Accept: application/soap+xml. Quant au corps du message celui-ci sera essentiellement de l'HTML commençant donc par

<?xml version="1.0" …>

et des parties contenant les balises soapenv:Enveloppe, soapenv:Body, etc. ainsi que des balises pour coder l'interrogation du personnage, balises qui ont été générées par la souche client.

Question 4 : La triche est une préoccupation forte des éditeurs de jeux en ligne.

4.1 Que peut apporter au jeu la mise en œuvre des propriétés de confidentialité, d'intégrité du flot de message, d'authentification, et de non-répudiation vis-à-vis des échanges de messages ? Justifiez brièvement votre point de vue. **(2 points)**

REPONSE

La triche est une préoccupation importante des éditeurs de jeux vidéo, et encore plus particulièrement pour les jeux vidéo multi-joueur en ligne.

*L'**intégrité du flot de messages** est essentielle, elle permet de se protéger contre toute manipulation des messages par les joueurs, ou par des personnes mal intentionnées. C'est une des deux propriétés incontournables peut-on raisonnable penser.*

*L'**authentification** est l'autre propriété essentielle. En effet, pour gérer les joueurs (gestion de profile, comptabilité, logique de jeu…) il faut qu'on soit sûr de leur identité, qu'ils soient effectivement "la personne qu'ils prétendent être".*

*La **confidentialité** permet de masquer le contenu des messages. Cette propriété n'a de sens que si il doit exister des interactions entre joueurs qui doivent rester secrètes. C'est le cas lorsque le jeu permet des alliances par exemple, et que des messages de coordination pour lancer une attaque par exemple sont nécessaires.*

*La **non-répudiation** est peut-être la moins évidente à utiliser dans un contexte multi-joueurs online. Toutefois, on peut imaginer dans un contexte d'univers de jeu virtuel où il existe une économie que les joueurs s'échangent des biens virtuels voire se les vendent. On peut alors avoir besoin d'un notaire électronique pour le jeu. Ce notaire offrant un enregistrement non discutable des transactions effectuées entre joueurs.*

4.2 On déploie une architecture à base de cryptographie à clef publique/clef secrète type RSA. Décrivez les étapes que devrait effectuer un protocole d'authentification du client auprès du serveur, on fait l'hypothèse que le client connaît déjà la clef publique du serveur. **(2 points)**

REPONSE

Le protocole d'authentification s'appuie sur des échanges de messages qui permettent à Bob (le client) de prouver à Alice (le serveur) qu'il est Bob :

1. ***Alice*** *génère un* ***nombre aléatoire***
2. ***Alice*** *envoie à* ***Bob*** *une requête d'authentification qui contient : [Alice, Bob,* ***nombre aléatoire****]*
3. ***Bob*** *reçoit la requête d'authentification*
4. ***Bob*** *produit* ***(Bob, nombre aléatoire)*** *et crypte le tout avec sa clef secrète qu'il est le seul à connaître, il envoie une réponse d'authentification à* ***Alice*** *:*
   [Bob, Alice, ***{chiffrement(Bob, nombre aléatoire)avec clef secrète de Bob}****]*
5. ***Alice*** *reçoit le message*
6. ***Alice*** *décrypte avec la* ***clef publique de Bob*** *qu'il détient déjà via un tiers de confiance (une autorité de certification par exemple):*
   {chiffrement(Bob, nombre aléatoire)avec clef secrète de Bob} *qui lui donne :* ***(Bob, nombre aléatoire)****, Alice peut alors déterminer que le nombre aléatoire est bien celui envoyé et que seul Bob est l'auteur de cet envoi car seul à posséder la clef secrète qui a permis le chiffrement initial*

4.3 Cela suffirait-il à authentifier le serveur vis-à-vis du client ? **(1 point)**

REPONSE

En l'état aucune information ne permet d'authentifier le serveur puisque aucune information que seule Alice connaîtrait n'est à la base d'un échange. Donc le serveur ne peut être authentifié par Bob.

Question 5 : On s'intéresse maintenant à des échanges de données de jeu via le courrier électronique. Après une interaction avec un personnage du jeu, afin de découvrir son identité, le joueur reçoit le message suivant :

Return-Path: <serveur-jeu@service-de-jeu.com>
X-Original-To: ben@cnam.fr
Delivered-To: ben@cnam.fr
Received: from node.cnam.fr (node.cnam.fr [163.173.128.20])
by node.cnam.fr (Postfix) with ESMTP id D0E5DEB172
for <ben@cnam.fr>; Thu, 22 Jun 2009 06:34:09 +0100 (CET)
Received: from mta1.service-de-jeu.com (mta1.service-de-jeu.com [195.167.228.242])
by node.cnam.fr (Postfix) with ESMTP id 3584B123AD0
for <ben@cnam.fr>; Thu, 22 Jun 2009 06:34:06 +0100 (CET)
Received: from service-de-jeu.com (localhost [127.0.0.1])
by mta1.service-de-jeu.com (8.12.9+Sun/8.12.9) with ESMTP id l285Y1Vg001884

for <ben@cnam.fr>; Thu, 22 Jun 2009 06:34:05 +0100 (MET)
Content-Transfer-Encoding: binary
Content-Type: multipart/alternative; boundary="_-la-cesure-=_1173332045212331209"
MIME-Version: 1.0
Organization: serveur-jeu@service-de-jeu.com
Reply-To: serveur-de-jeu@service-de-jeu.com
X-Comment: CODEOP=20090622-1-39,QOS=1,ID=Z2VyYXJkQGNuYW0uZnI=
X-Mailer: mailing.1.8.0 by FG
Date: Thu, 22 June 2009 05:34:05 UT
From: serveur-jeu@service-de-jeu.com
To: ben@cnam.fr
Subject: =?ISO-8859-1?Q?rencontre d'un mage elfe ?=
Message-Id: <1173332045.Z2VyYXJkQGNuYW0uZnI=.1.20090622-1-39@service-de-jeu.com>

This is a multi-part message in MIME format.
--_-la-cesure-=_1173332045212331209
Content-Disposition: inline
Content-Length: 198
Content-Transfer-Encoding: 8bit
Content-Type: text/plain; charset="iso-8859-1"
MIME-Version: 1.0
Date: Thu, 22 June 2009 05:34:05 UT

Vous avez vu lou un elfe, identificateur 0016, du royaume mforet et de suzerain elroon, il a le pouvoir d'un mage.

……le mail se poursuit mais la suite n'est pas intéressante pour l'exercice……

5.1 On trouve au début du courrier plusieurs lignes d'informations commençant par '**received**'. A quoi sert cette information ? Retracer l'histoire de l'acheminement de ce courrier (lieux par lesquels il est passé et heures) ? **(1 point)**

REPONSE

La réponse à cette question amène donc à regarder les entêtes 'received' successives dans l'ordre inverse ou elles apparaissent dans le message pour reconstituer l'histoire de l'acheminement du message. On pourra aussi utiliser les heures de passage dans les différents relais de courrier.

a) **Received:** from service-de-jeu.com (localhost [127.0.0.1]) by mta1.service-de-jeu.com (8.12.9+Sun/8.12.9) with ESMTP id l285Y1Vg001884 for <ben@cnam.fr>; Thu, 22 Jun 2009 06:34:05 +0100 (MET)

Le message a été émis par un client de messagerie d'un hôte appartenant à un site de jeu dont le nom de domaine est service-de-jeu.com. Il a été relayé dans le domaine service-de-jeu.com par un serveur de messagerie de ce domaine dont le

nom est mta1.service-de-jeu.com. L'heure de passage a été 6h34 05s.

b) **Received:** from mta1.service-de-jeu.com (mta1.service-de-jeu.com [195.167.228.242]) by node.cnam.fr (Postfix) with ESMTP id 3584B123AD0 for <ben@cnam.fr>; Thu, 22 Jun 2009 06:34:06 +0100 (CET)

Le message a été transmis ensuite par mta1.service-de-jeu.com au serveur de messagerie du CNAM qui s'appelle node.cnam.fr. Au passage nous apprenons que l'applicatif de serveur de messagerie retenu par le CNAM est postfix. Le message est arrivé à 6h34 06s.

Remarque : L'horloge n'est pas nécessairement synchronisée avec la précédente. Peut être utilisent-elles NTP Network Time Protocol toutes les deux mais la qualité de la synchro NTP n'est pas excellente. Donc la durée d'acheminement entre mta1.serveur-de-jeu.com et node.cnam.fr qui serait de l'ordre de 2 secondes traduit peut être autant la qualité de la synchronisation entre les deux machines qu'un délai très précis.

e) **Received**: from node.cnam.fr (node.cnam.fr [163.173.128.20]) by node.cnam.fr (Postfix) with ESMTP id D0E5DEB172 for <ben@cnam.fr>; Thu, 22 Jun 2009 06:34:09 +0100 (CET)

Le message a maintenant été relayé une dernière fois entre le serveur node.cnam.fr du CNAM dont et lui-même. Comme il s'agit du même serveur de courrier on peut faire l'hypothèse qu'il y a un traitement en local qui n'est pas spécifié (anti-virus, anti-spam…).

Comme c'est le dernier serveur qui a été visité par le message c'est donc sur ce serveur que la relève du courrier à été faite pour être affiché à son destinataire par le client de messagerie de ce destinataire. Bien que ça ne soit pas indiqué dans le courrier, il est tout à fait probable que ce serveur exécute un logiciel MDA ('Mail Delvery Agent') c'est-à-dire un serveur POP ou un serveur IMAP qui interroge des boites aux lettres pour le compte des clients de messagerie.

5.2 Expliquez la directive suivante. **(1 point)**

Subject: =?ISO-8859-1?Q?rencontre d'un mage elfe ?=
Indications : Quel est son objectif ? Pourquoi cette forme de présentation bizarre ?

REPONSE

Il s'agit de la ligne d'entête qui définit, le sujet dans un courrier électronique (mot-clef' subject'). Ce sujet concerne le jeu dont on se préoccupe et indique qu'un "mage elfe" a été rencontré par le joueur au cours de son périple.

Sa représentation est curieuse. Pourquoi cette forme avec des points d'interrogation etc … ?

Le problème avec les courriers RFC 822 est que tout doit passer sous la forme de caractères US-ASCII sept bits. Or nous utilisons le français, dans le sujet du courrier électronique, on peut rencontrer des signes diacritiques que sont l'accent grave sur le a et l'accent aigu sur le e. Pour arriver à présenter correctement un texte en français du sujet du courrier, il faut permettre un codage pour ces caractères spéciaux n'existant pas en anglais.

Pour cela, un artifice est utilisé, il consiste à encadrer ce texte qui est donc défini dans un alphabet différent de l'USASCII, par deux délimiteurs = ? et ?=. A l'intérieur de ces délimiteurs, on va définir trois informations séparées par des points d'interrogation.

Le premier champ indique qu'on va utiliser l'alphabet ISO-8859-1 encore appelé alphabet Latin-1. C'est un alphabet sur 8 bits qui permet d'exprimer l'essentiel des caractères des langues européennes occidentales : langues latines dont le français mais aussi nordiques avec l'allemand et aussi l'anglais. C'est pourquoi ISO latin 1 apparaît très fréquemment comme un compromis acceptable dans de nombreuses normalisations. Il existe d'autres alphabets comme latin 2 pour les langues slaves etc ….

Le Q dans le second champ signifie 'Q encoding'. Ce champ définit le mode de codage qu'on va utiliser pour le sujet du message situé dans le troisième champ. 'Q encoding' est un codage très voisin pour le sujet des messages du codage 'quoted printable', qui est un codage pour les textes dans les corps des messages. 'Quoted printable' est donc une façon normalisée par MIME pour coder des suites d'octets d'un alphabet quelconque en caractères de l'alphabet US ASCII 7 bits. Ici, ce format d'encodage va encoder des caractères de l'alphabet latin 1 formant le texte du sujet. En fait on n'utilise pas tout à fait la même variante de l'encodage quoted printable dans les sujets de courriers (avec le Q encoding) que dans les contenus de courrier.