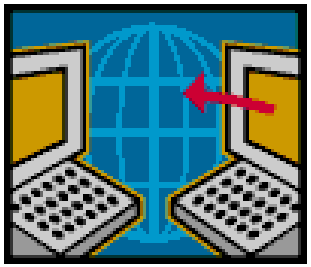
Durée prévue 4h30

Phase 1 : requête du méta file

1.1 Client

Programme client_web.c
Envoi d'une requête sur le port 7800
''Get /audio.ram HTTP/1.1\r\n''



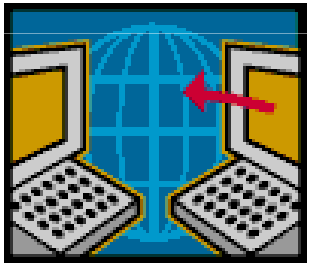
1.2 Serveur web

Programme : serveur_web.c
A l'écoute sur le port 7800
Renvoie le meta file demandé
Contenu du meta file :
http://localhost:7890/bbc.wav
On utilise http à la place de rtsp (virtual streaming)

Phase 2 : requête du media file et lecture du flux

2.1 Client

Programme : client_audio.c
Envoi d'une requête sur le port 7890
de localhost
''Get /audio.wav HTTP/1.1\r\n''



2.2 Serveur de flux audio

Programme : serveur_audio.c
A l'écoute sur le port 7890
Renvoie le media file (bbc.wav) demandé
- recherche du fichier bbc.wav sur le disque
- lecture de l'entête
- lecture des données
- envoie les données sous la forme d'un flux raw

2.3

Lecture des données reçues (streaming)
Envoie le flux de données au fur et à mesure
sur les haut-parleurs (vers /dev/dsp)

2.4 (vu en cours)

Contrôle du serveur : stop/replay sur un autre port. Gestion du buffer tampon de lecture.
(simulation de rtsp)

Documents du projet

Ouverture et lecture d'un fichier wav. Envoie du flux dans les HP

La gestion audio sous linux est assez complexe. Cette partie est donc implémentée dans le serveur et le client audio

Pour tester vos HP (sortie casque) à partir d'un fichier wav :
sox bbc.wav –t ossdsp /dev/dsp

Pour accéder au périphérique /dev/dsp, utilisez padsp. Il s'utilise en spécifiant en paramètre la commande qui veut un accès au périphérique
Ex: padsp ./client_audio localhost *http://localhost:7890/bbc.wav*

Pour envoyer un flux dans un haut parleur :
status=write(fdson,buf,sizeof(buf)); // voir aussi le programme loadWave.c
Avec fdson=open(''/dev/dsp'',O_RDWR);

Documents du projet (suite)

Lecture et écriture de données à partir d'une socket

Il existe plusieurs méthodes suivant les données :

(voir document page 7)
- send et recv (utilisables pour des données textes)
- send_string et recv_line (plus simples d'emploi et plus évolués)

Pour le flux binaire :

```
Int sockfd; FILE *fdesc; FILE *fd; size_t  n;
char buf[LINELENGTH];
……..                                   /* Mise en place de la connexion (voir par ex client_audio)
……..
fdesc=fdopen(sockfd,''r'');             /* sockfd est la socket */
.....
while ((n=fread(buf,sizeof(char),LINELENGTH,fdesc)))  /* fdsec sert de buffer intermédiaire pour la
 {                                                      socket */
        status=write(fdson,buf,LINELENGTH);          /* On envoie le flux dans les HP*/
        status=ioctl(fdson,SOUND_PCM_SYNC,0);
      if (status==-1) perror(…)
}
```