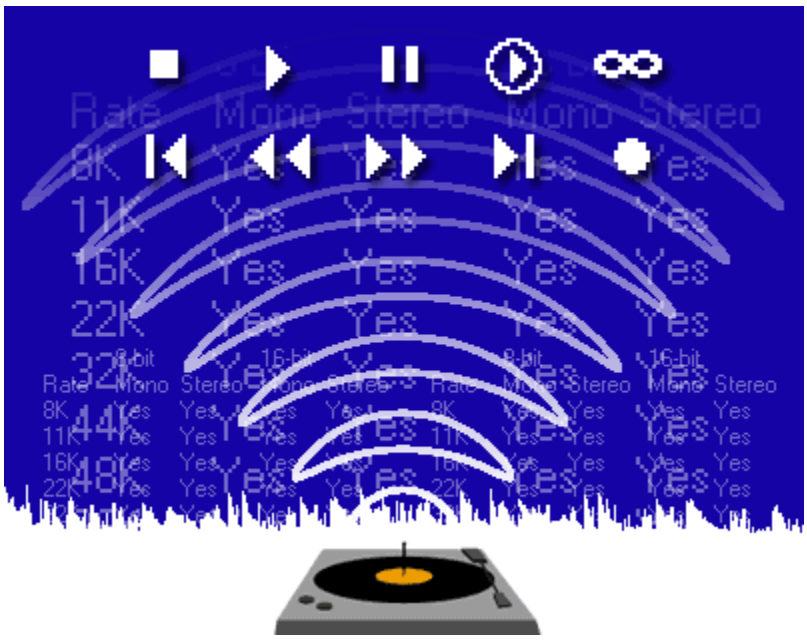
Music Server

Ονοματεπώνυμο Α.Μ.	Ψαλλίδας Φώτης 1115200600170
Μάθημα	Προγραμματισμός Συστήματος
Αντικείμενο Εργασίας	Υλοποίηση Audio Server
Καθηγητής	κ. Αλέξης Δελής
Ακαδημαικό Έτος	2008-2009

Περιεχόμενα



Αρχεία-Γενικές Πληροφορίες

	Lines	Words	Characters	Page Reference
url_codec.hpp	**32**	**224**	**1232**	4
url_codec.cpp	**144**	**488**	**2855**	4
threadf.hpp	**57**	**138**	**1089**	4
test.sh	**402**	**2401**	**31253**	14
tcp.hpp	**81**	**261**	**2275**	5
tcp.cpp	**127**	**429**	**3365**	5
str.hpp	**45**	**153**	**1094**	6
str.cpp	**200**	**903**	**6444**	6
path.hpp	**243**	**888**	**5868**	7
musicserver.cpp	**619**	**2297**	**20123**	12
Makefile	**69**	**238**	**1730**	13
http.hpp	**113**	**405**	**4174**	11
http.cpp	**747**	**2502**	**20387**	11
htree.hpp	**124**	**435**	**5644**	8 – 9
htree.cpp	**779**	**3015**	**25605**	10
flist.hpp	**114**	**370**	**2480**	10
data.hpp	**29**	**63**	**523**	10
data.cpp	**123**	**422**	**3446**	10
400.html	**49**	**199**	**2363**	11
404.html	**49**	**194**	**2279**	11
505.html	**47**	**196**	**2317**	11
Total	**4193**	**16211**	**146546**	

Επεξήγηση-Λειτουργικότητα Αρχείων

url_codec.hpp (credits to Spiros Evangelatos)

Header file για τον ορισμό των ρουτινών κωδικοποίησης και αποκωδικοποίσης των urls.

url_codec.cpp (credits to Spiros Evangelatos)

Source code για την κωδικοποιήση και αποκωδικοποίση των urls.
Χρησιμοποιούνται για την αποκωδικοποίση του get url που ζητάει ο χρήστης μέσα από το http requst που στέλνει.Επίσης χρησιμοποείται κατα την αποστολή playlist όπου τα τραγούδια έχουν την μορφή http://server:port/encoded_path

threadf.hpp

Ορισμός κλάσσης για την αποθήκευση των δεδομένεων που χρησιμοποιεί το κάθε thread,έτσι ώστε να ειναι visible από τα υπόλοιπα threads και να μπορεί να διαγραφτεί από αυτά.Έτσι αποφεύγουμε memory leaks κατα το exit.

tcp.hpp

Ορισμός κλάσσης διαχείρησης του TCP/IP protocol για την επικοινωνία
μεταξύ server και clients.Αναλυτικα στην κλάσση αυτή ορίζονται οι συναρτήσεις:

TCP usage API routine	Net functions' usage to implement TCP api	Explanation
TCP::i_open()	**socket() , (setsockopt()) , bind() ,listen()**	• Δημιουργεί το socket • Κάνει bind σε μία συγκεκριμένη port και • Δημιουργεί μια ουρά μήκους "lqlen" για τις εισερχόμενες αιτήσεις των clients. Το socket που δημιουργείται εχει domain AF_INET.
TCP::i_accept()	**accept()**	Διαβάζει την ουρά που δημιούργησε η listen ώστε να δεχτεί μια νέα σύνδεση. Ενημερώνει το new_sock το οποίο ειναι το fd που μπορεί να χρησιμοποεί ο server με τον client για την μεταξύ τους επικοινωνία. Επιστρέφει 1σε επιτυχία -1 σε αποτυχία.
TCP::i_getsockname()	**getsockname()**	Χρησιμοποείται για την εύρεση της ip και της port με τις οποίες σεμμετέχει ο server στην επικοινωνία με τον client, που η αίτηση του έγινε accepted.

tcp.cpp

Source code για την υλοποίηση των συναρτήσεων που έχουν ορισθεί στο tcp.hpp
και εξηγούνται από πάνω. Ακόμη υλοποιούνται και κάποιοι accessors για private data της κλάσσης **TCP**.

str.hpp

Ορισμός χρήσιμων συναρτήσεων για την διαχείρηση strings καθώς και οπτικοποίση του intro στον Music Server.

Routine	Explanation
std::string remove_white(std::string)	Remove whitepsaces at statrt and at the end of a string
std::string remove_slash(std::string)	Remove multiple slashes from a string
int mystrcmp(const char *, const char *)	Case insensitive strcmp
char* mystrstr(const char *, const char *)	Case insensitive strstr based on Knuth-Moris-Pratt algorithm
template <class T> bool from_string(T& t, const std::string& s, std::ios_base& (*f)(std::ios_base&))	Converting strings to numbers Usage of templates to pass the type of number.
template <class T> std::string to_string (const T& t)	Convert numbers to strings
void print_intro(void)	Clearing screen(if possible), Printing graphically intro.

str.cpp

Source code για την υλοποίηση των συναρτήσεων του str.hpp
(εκτος των from_string και to_string που υλοποιούνται μέσα στο str.hpp λόγω templates)

path.hpp

Header file με χρήσιμες συναρτήσεις για την μετατροπή μοναπατιών απο relative σε absolute και εύρεση του τρέχοντος καταλόγου.

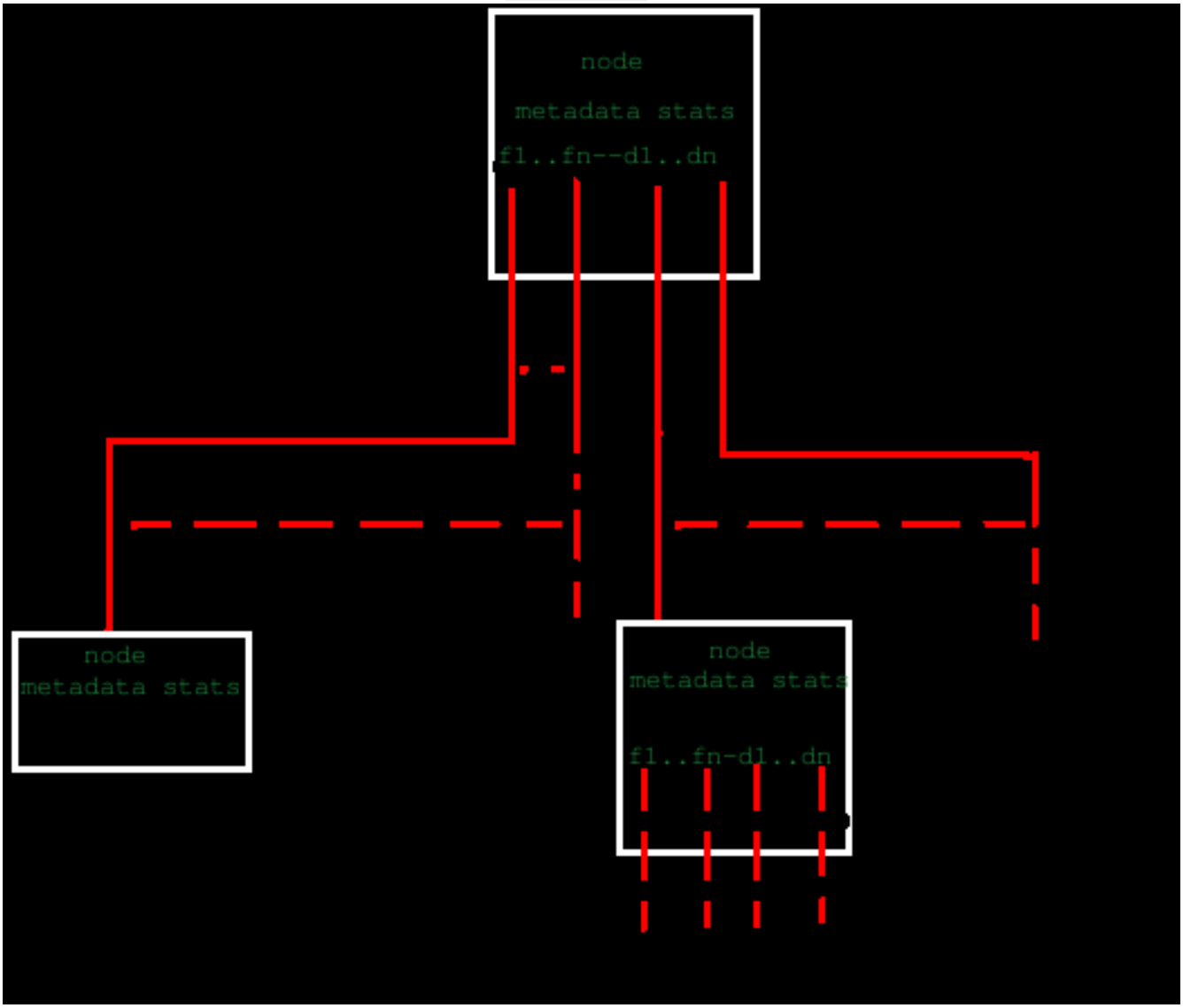
Routine	Explanation
char* fix_path(const char *path, const char *cwd)	Creates the absolute path of a file. For symbolik links this routine finds where it points and returns the path of the file-directory it points to. In case of circles error occurs.
char* get_cwd()	Get current working directory.

Η συνάρτηση fix_path χρησιμοποείται για την δημιουργία του απόλυτου path του musicdir που δίνει ο χρήστης για τον audio directory.Αυτο γίνεται διότι ο έλγχος μεταφέρεται στον κατάλογο αυτό και η συνάρτηση chdir δεν έχει τη δυνατότητα να μεταφερθεί σε path με relative format.
Η συνάρτηση get_cwd() χρησιμοποείται για να αποθηκεύσουμε το path του τρέχοντα καταλόγου ώστε να τον περάσουμε σαν όρισμα στην fix_path.

htree.hpp

Header file για τον ορισμό της κλάσσης HTREE η οποία χρησιμοποείται για την αποθήκευση των paths των αρχείων σε μια ιεραρχική μορφή
(όμοια με αυτή του filesystem σίγουρα όμως πιο ελαφριά,διότι δεν κρατάει τόσα δεδομένα όσο καρατάει και το σύστημα).

Virtualize



Για καθε directory ή file δεσμέυεται ένας κόμβος.Στον κόμβο αυτό υπάρχουν κάποια στατικά δεδομένα και κάποια μεταδεδομένα για το directory ή το file. Αν ο κόμβος ειναι κόμβος directory τότε υπάρχει μια λίστα με όλα τα files και τα subdirectories που περιέχει αυτός.
Ετσι όταν ο χρήστης-client ζητήσει ένα συγκεκριμένο path τότε αυτό το path αναλύεται σε κομμάτια αποτελούμενα από τα directories της ιεραρχίας καθώς και το όνομα του file.Ετσι η αναζήτηση γίνεται σε O(n) όπου n το πλήθος των directories +1 (λόγω του file) στο requested path συν την όποια αναζήτηση μέσα στις λίστες (max O(k) όπου k το μέγεθος της λίστας).Μια βελτίωση της παρούσας κατάστασης θα ήταν η χρήση hash_set αντι λίστας έτσι ώστε να μειωθεί το O(k) σε O(1) {δεδομένου ότι κάθε αρχείο έχει μοναδικό name μέσα στο directory θα έχει και μοναδικό hash_value αν δημιουργεί μια hash function που να χρησιμοποιεί αυτη τη μοναδικότητα του filename}.

<table>
<tr><th>Routine</th><th>Explanation</th></tr>
<tr><td>HTREE::htree()</td><td>Htree constructor</td></tr>
<tr><td>HTREE:~htree()</td><td>Htree destructor</td></tr>
<tr><td>bool HTREE::match_extension(std::string music,struct metadata**)</td><td>Used to determine whether a file is an audio file and combine the audio file with its content/type.The audio files supported are :
<table>
<tr><th>Extension</th><th>Content-Type</th></tr>
<tr><td>Mp3</td><td>mpeg</td></tr>
<tr><td>Ogg</td><td>ogg</td></tr>
<tr><td>Aac</td><td>aac</td></tr>
<tr><td>Wma</td><td>x-ms-wma</td></tr>
<tr><td>M4a</td><td>mp4a-latm</td></tr>
<tr><td>M4p</td><td>mp4a-latm</td></tr>
<tr><td>Flac</td><td>x-flac</td></tr>
</table></td></tr>
<tr><td>i_node* HTREE::insert_under(i_node*,i_node*)</td><td>Inserting a new node in its parent list.</td></tr>
<tr><td>bool HTREE::upload(struct metadata*,i_node*)</td><td>Trace filesystem under MusicDir and
upload i_nodes in htree properly.</td></tr>
<tr><td>FList<i_node>* HTREE::songsearch(std::string)</td><td>Searching all audio files in htree
and check if they match a pattern.
All audio files' metadata which match the pattern are uploaded in a list
which gets returned.</td></tr>
<tr><td>struct metadata * HTREE::pathsearch(std::string)</td><td>Search a certain song in MusicDir
using htree.</td></tr>
<tr><td>i_node* HTREE::finds(i_node*,std::string,std::string,FList<i_node>*,bool)</td><td>Tree search used by songsearch.</td></tr>
<tr><td>i_node* HTREE:findp(i_node*,std::string,std::string,struct metadata**,bool)</td><td>Tree search used by pathsearch.</td></tr>
<tr><td>void HTREE::findall(i_node*,FList<i_node>*)</td><td>Returns a list with all audios under MusicDir.</td></tr>
<tr><td>i_node* HTREE::print_tree(i_node*,int)
i_node* HTREE::print()</td><td>Print tree hierarchy and some statistics in a graphic-way (looks like tree's command output)</td></tr>
</table>

htree.cpp

Source code για την υλοποίση των συναρτήσεων του i_node και του HTREE όπως αυτές ορίζονται στο htree.hpp.

flist.hpp

Header file για την υλοποίηση templated λιστας η οποία χρησιμοποείται σε πολλά σημεία της εφαρμογής.Επιλέχθηκε να ειναι templated για αυτον ακριβώς το σκοπό αφου τα δεδόμενα των διαφόρων λιστών που χρησιμοποεί το σύστημα ποικίλλουν.

data.hpp

Header file για την ορισμό των μεταδεδομένων και τη συναρτήσεις δημιουργίας τους
Αναλυτικά τα δεδομένα που κρατάει το struct των μεταδεδομένων:

Struct metadata	Explanation
char* path_name	absolute path to file-directory
char* name	filename-dirname
char* rights	Privilleges on file's-directory's usage for different users
char* audio_type	The content_type of audio file
mode_t modes	Modes for directory-file taken by stat (st_modes field)
int is_dir	Flag showing if metadata is about directory or file
time_t c_time	Creation time

Οι συναρτήσεις δημιουργίας χρησιμοποιούνται κατα την δημιουργία του δέντρου (htree) ώστε να αποθηκεύονται στους κόμβους σωστά μεταδεδομένα.
Αναλυτικά οι συναρτήσεις δημιουργίας μεταδεδομένων:

Routine	Explanation
struct metadata*create_metadata(char*,char*,char*,mode_t,int,time_t)	Allocate memory in heap for new metadata and paste proper data in it.Returns a pointer to newly allocated meatadata
struct metadata* take_metadata(char*,char*)	Take the name of a <file | dir> and mins data from filesystem's info which uses to create its metadata and return a pointer to it.

data.cpp

Source code για την υλποίηση των συναρτήσεων που ορίζονται στο data.hpp και περιγράφονται από πάνω

400.html

Αρχείο html που αποστέλλεται στον client σε περίπτωση Bad request.

404.html

Αρχείο html που αποστέλλεται στον client σε περίπτωση που δεν βρέθηκε αυτό που ζητούσε (Not Found).

505.html

Αρχειο html που αποστέλλεται στον client σε περίπτωση Inetrall Error του server.

http.hpp

Ορισμός κλάσεων HTTP_H και header και των αντίστοιχων συναρτήσεων διαχείρησης. Objects της HTTP_H κλάσης εχουν τη δυνατότητα ανάγνωσης των HTTP requests που στέλνει ο client αλλά και την αντίστοιχη δυνατότητα επεξεργασίας της αίτησης προβαίνωντας σε κατάλληλες ενέργεις και τέλος αποστολή μιας συγκεκριμένης απάντησης, που αποτελεί αποτέλεσμα της επεξεργασίας του HTTP request.

Routine	Explanation
HTTP_H::HTTP_H(int,char*,char*)	Constructor
HTTP_H::HTTP_H(int,char*,char*,std::string*,char*)	Another constructor
~HTTP_H::HTTP_H()	Destructor
int HTTP_H::read()	Read an HTTP request,find the requested url create the list with all headers received. In case of error send error responce.
void HTTP_H::writesong(struct metadata*)	Reply to client sending back the requested song.
void HTTP_H::writeplaylist(FList<htree::i_node>*,int)	Reply to client sending back a list of songs which match the requested pattern, in format http://server:port/encoded_path.
void HTTP_H::senderror(int)	Routine used to send proper error back to client. 400: Bad request 404: Not Found 505: Internal Error
std::string HTTP_H::make_response(int,std::string,unsigned long,std::string)	Routine used to create a string with the initial line and the headers of the response.
std::string HTTP_H::get_url()	Accessor
int HTTP_H::get_stype()	Accessor
void HTTP_H::print()	Print several infos for client's request.
void HTTP_H::strp(std::string,int)	Printing to stdout or stderr a string in specified format

http.cpp

Source code υλοποίησης των συναρτήσεων που ορίζονται στο htree.hpp και περιγράφονται παραπάνω.

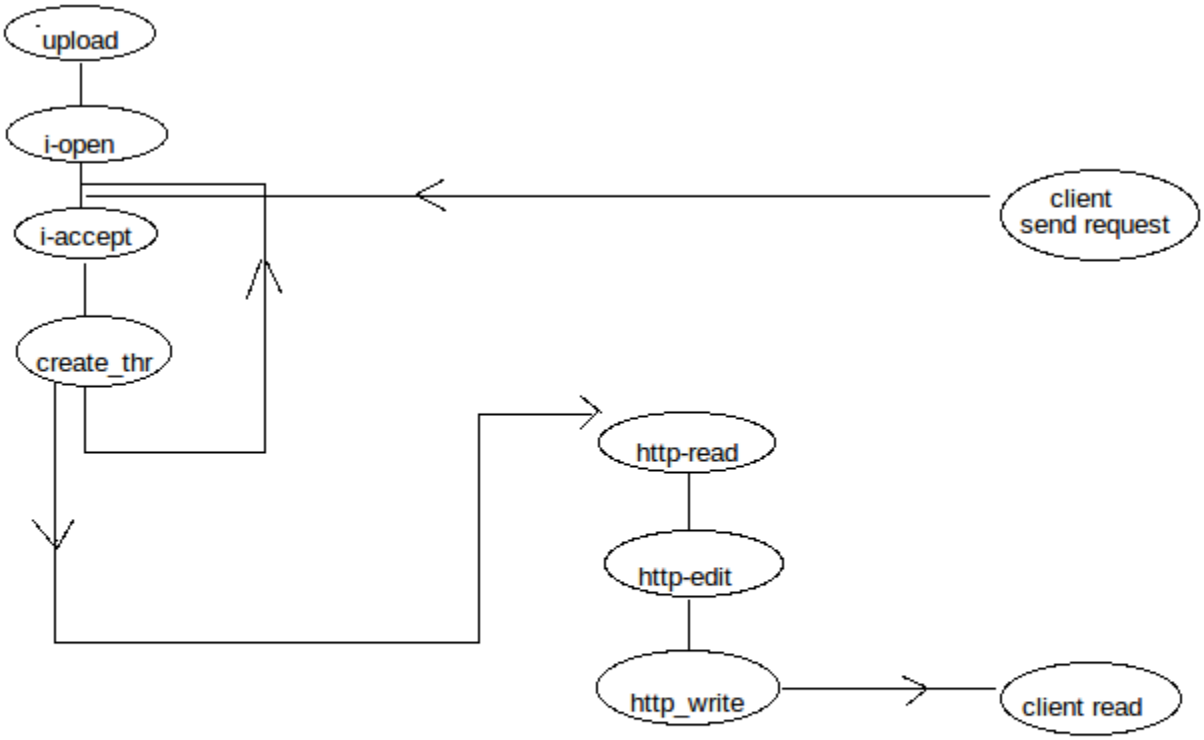
musicserver.cpp

Main κώδικας για τον Music Server. Σε αυτό το αρχείο υλοποίειται η όλη διαδικασία που εκτελεί ο server για την επικοινωνία του με τους διάφορους clients.

Αρχικά ο server ανεβάζει στη μνήμη ολα τα paths των αρχείων που εντοπίστηκαν στο Music Directory όπως αναλύθηκε παραπάνω.

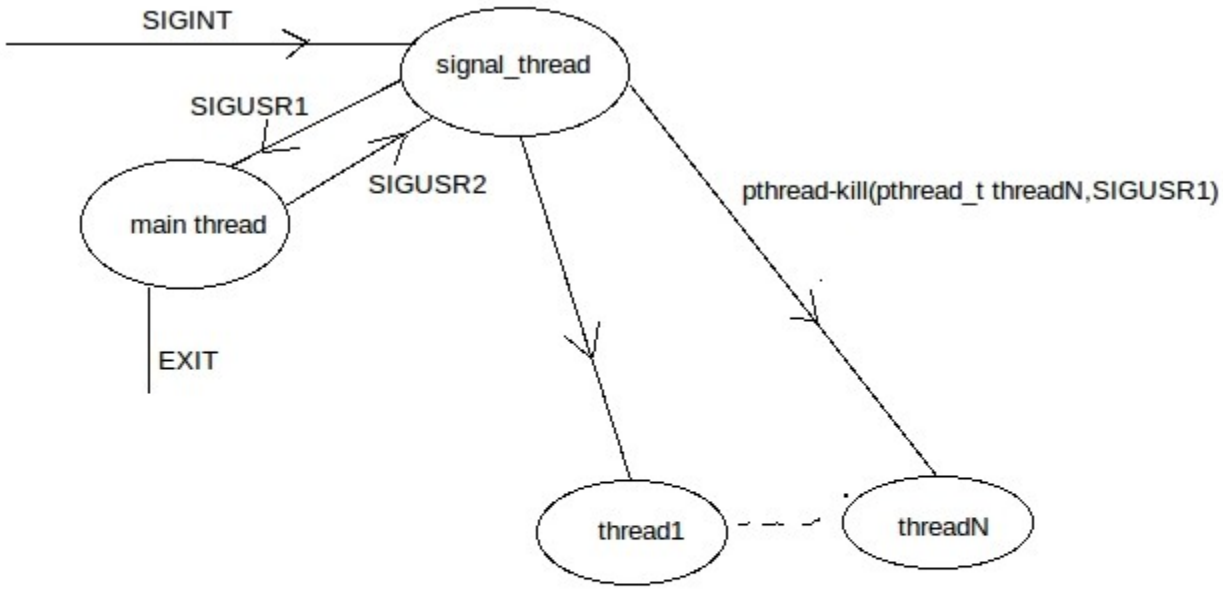
Στη συνέχεια ο server χρησιμοποιώντας object της κλάσης TCP δημιουργεί ένα νεο socket το κάνει bind σε μια address και δημιουργεί μια ουρά για τις αιτήσεις των clients (i_open). Μετά κάνει i_accept() περιμένωντας αιτήσεις από clients.Για κάθε νεα αιτηση ο client δημιουργείται ενα νεο thread το οποιο χρησιμοποιώντας ένα object της HTTP_H κλάσης μπορεί να διαβάσει ,να επεξεργαστει και να απαντήσει το request του client.

Συνολικά η όλη διαδικασία που εκτελει ο server έχει ως εξής.



Για να ειναι ποιο safe η εφαρμογή απο άποψη μνήμης χρησιμοποείται μια τεχνική με signals. Το main thread του server δημιουργεί ενα thread το οποίο απλά περιμένει να λάβει signal SIGINT. Ολα τα άλλα threads κάνουν block το SIGINT ετσι αν γινει rise θα το λάβει μόνο το συγκεκριμένο thread.Οταν το λάβει καταστρέφει όλα τα threads διαγράφωντας και όλη τη μνήμη που χρησιμοποιουν (εχει πρόσβαση στη λίστα που κρατάει το thread_f objects).Επειτα ο έλεγχος πειστρέφεται στο main thread ώστε να σκοτώσει και το signal thread και έπειτα να κάνει exit.

Η όλη διαδικασία από άποψη σημάτων έχει ως εξής:



Makefile

Αυτοματοποιημένη μεταγλώτιση όλων των αρχείων που συνεργάζονται για τη δημιουργία του server.Καλό θα είναι να μεταγλωτίσετε τα αρχεία στον spiderman ή τουλάχιστον να χρησιμοποιήσετε την εκδοχή του make που υπάρχει σε αυτον τον υπολογιστή για να μην σας φανεί άσχημο το ouput..(Χρησιμοποείται @echo -e).
Αν θέλετε να μεταγλωστίσετε τα αρχεία σε Sun Workstation στο di.uoa.gr domain πρέπει να πειράξετε το makefile και να αφαιρέσετε τα σχολια από τις γραμμές που λεει στην αρχή του Makefile.
Το τελικο εκτελέσιμο που παράγεται ειναι το musicserver και το οποίο μποείτε να χρησιμοποιήσετε για να εκκινήσετε τον server.

./musicserver -p <PORTNUM> -d <MUSICDIR>
ή
./musicserver -d <MUSICDIR> -p <PORTNUM>

Αν θέλετε κάποιο help πάνω στον musicserver μπορειτε να πατήσετε

./musicserver -h
ή
./musicserver –help

test.sh

Bash scriptακι για το testing του καταλόγου.Πρεπει να καλεστει ώς

./test.sh -p <PORTNUM> -s <SERVER>

όπου portnum ή port που ακούει ο server και SERVER καλό ειναι να δώσετε 195.134.66.107 και να τρέχετε τον server στο spiderman.di.uoa.gr.

Αυτο που κάνει το scriptaki ειναι να βλέπει ποιά απο τα {wget,firefox,totem,amarok} υπάρχουν εγκατεστημένα στο PC και αντιστοιχα να ρωτάει αν θέλετε να χρησιμοποιηθεί για το testing του server.

Ακολούθως ρωτάει αν θέλετε να κατεβούν-αναπαραχθουν όλα τα τραγούδια χρησιμοποιώντας την εφαρμογή(ή τις εφαρμογές) που επιλέχθηκε-αν ή να χρησιμοποιηθει η εφαρμογή για ένα μόνο request.

Τελος κάνει αυτό που του ειπατε.

Σε περίπτωση που δεν υπάρχει κάποια εφαρμογή ρωτάει αν θέλετε να τη κατεβάσετε.

Πρεπει να έχετε δικαιώματα sudo.

ΛΟΙΠΕΣ ΛΕΠΤΟΜΕΡΕΙΕΣ

Ο Music Server δοκιμάσθηκε με τα προγράμματα wget amarok firefox (me plugin mplayer) καθως και με το totem. Αναπτύχθηκε σε περιβάλλον ubuntu 9.04 (jaunty jackalope) με τη χρήση του κειμενογράφου gedit.

Οι αναζητήσεις που μπορούν να γίνουν έχουν ως εξης.

Αναζήτηση τραγουδιού:

http://server:port/path

Όπου το path είναι το path κάτω από το κατάλογο Music Directory

Για παράδειγμα αν Musicidr=/home/Musicdir και αναζήτουμε το τραγούδι /home/Musicdir/song.mp3 τότε το path πρέπει να είναι song.mp3

Η αναζητηση στηρίζετε σε case insensitive matching οποτε μπορείτε να δώσετε ακομα και SoNg.mP3 για παράδειγμα.

Αναζήτηση Playlist:

http://server:port/songsearch/pattern.m3u

Οπου pattern μπορεί να είναι το οτιδήποτε και μπορεί να κάνει match με κάποιο filename.

Αν θέλετε την playlist με όλα τα τραγούδια πρέπει να δώσετε

http://server:port/songsearch.m3u