

Cypher System
Sistemes Operatius
La Salle - Universitat Ramón Llull
Nom: Víctor Garrido Martínez
Data: 20.06.2020

Índex

1	<i>Disseny</i>	3
1.1	Diagrames	3
1.1.1	Exemple Bàsic de funcionament del sistema	3
1.1.2	Diagrama General Del Servidor	4
1.1.3	Connect.....	4
1.1.4	Show Connections	4
1.1.5	Say Client	5
1.1.6	Say Server.....	5
1.1.7	Broadcast	5
1.1.8	Show Audios	5
1.1.9	Download Audio Server	6
1.1.10	Download Audio Client	6
1.2	Estructures de dades	7
1.2.1	Configurtion	7
1.2.2	Connection.....	7
1.2.3	Llista amb punt d'interès dinàmica	8
1.3	Recursos del sistema utilitzats	9
1.3.1	Signals	9
1.3.2	Pipes	9
1.3.3	Exclusió mútua	9
2	<i>Problemes Observats</i>	10
3	<i>Estimació temporal</i>	12
4	<i>Conclusions i propostes de millora</i>	13
5	<i>Bibliografia</i>	14

1 Disseny

1.1 Diagrames

Per mostrar el funcionament, anem a suposar que tenim 4 processos trinity funcionant:

1.1.1 Exemple Bàsic de funcionament del sistema

1. Persefone al port 8220
2. Niobe al port 8221
3. Merovingio al port 8222
4. Link al port 8223

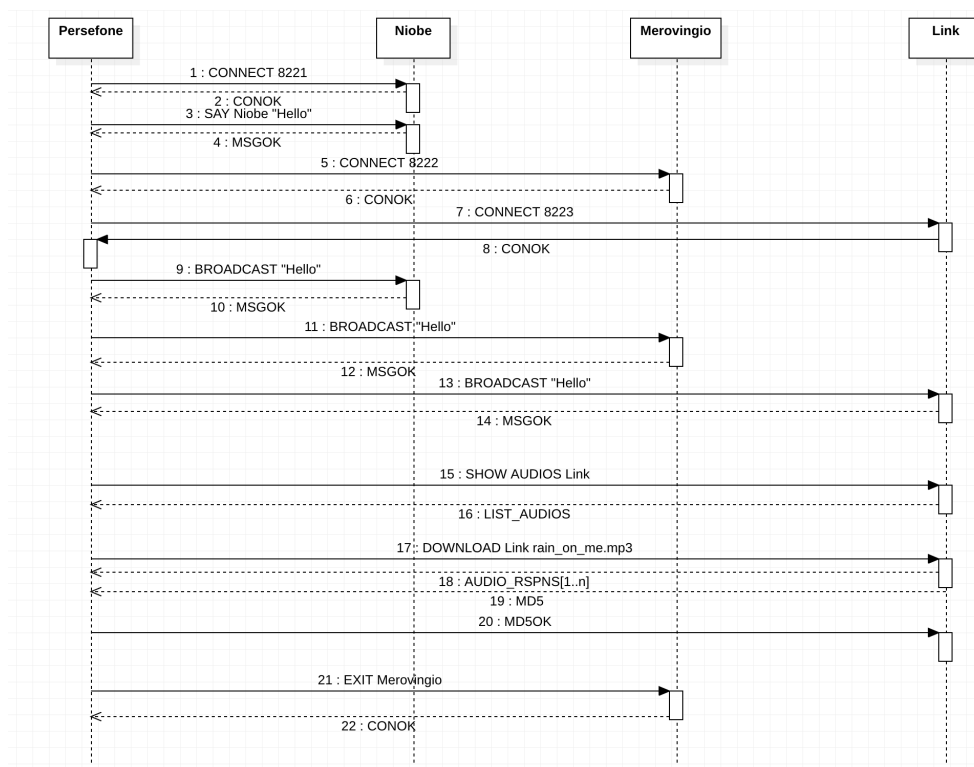


Figura 1: Diagrama de Seqüència General

Com es pot veure, tot el sistema es basa en un protocol de peticions i respostes.

1.1.2 Diagrama General Del Servidor

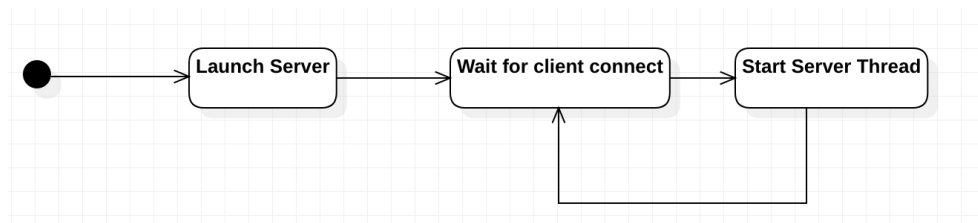


Figura 2: Diagrama d'Activitats General Del Servidor

La dinàmica del servidor es senzilla: primer el crea, i per cada nova connexió amb un client obre un thread dedicat que s'encarrega de manejar les peticions del client en qüestió.

Cada server thread, llegirà el header de la petició, i executarà les funcions descrites a continuació:

1.1.3 Connect

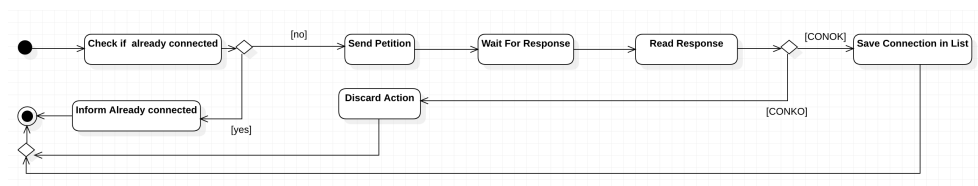


Figura 3: Diagrama Connect

1.1.4 Show Connections

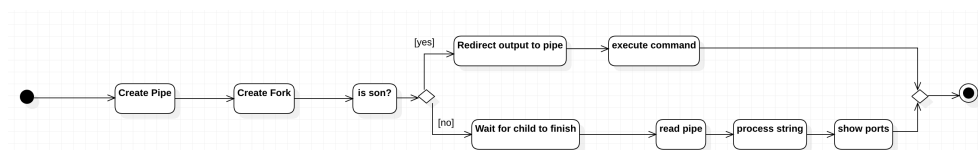


Figura 4: Diagrama Show Connections

1.1.5 Say Client

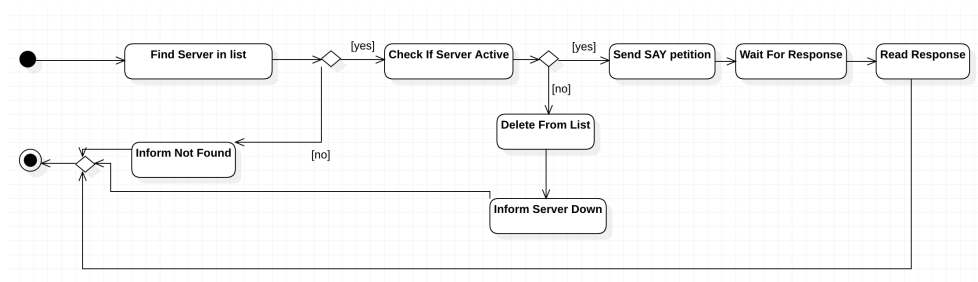


Figura 5: Diagrama Say Client

1.1.6 Say Server

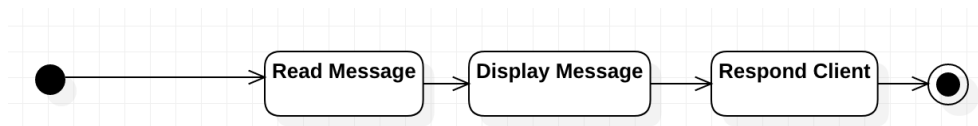


Figura 6: Diagrama Say Server

1.1.7 Broadcast

En aquest cas, segueix la mateixa mecànica que amb el SAY. La diferència recau en que el client recorre la llista de servers per tal d'enviar cada una de les peticions de missatge i que printa per pantalla l'ack de cada server.

1.1.8 Show Audios

En quant al client, segueix el protocol per enviar peticions, ja vist en el 1.1.5. Comprova que està connectat al server, comprova que està viu i envia la petició, amb la única diferència que la petició és SHOW AUDIOS

En quant al server, simplement llegeix els fitxers i retorna el llistat.

1.1.9 Download Audio Server

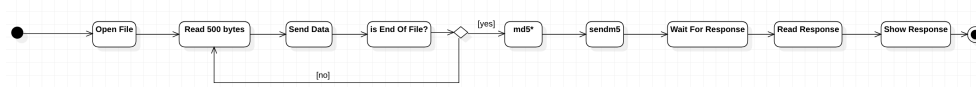


Figura 7: Diagrama Download Audio Server

1.1.10 Download Audio Client

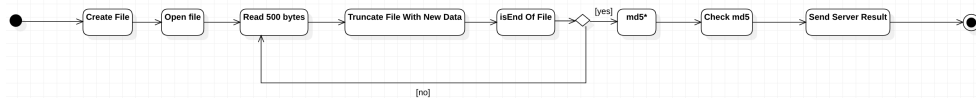


Figura 8: Diagrama Download Audio Client

Cal destacar que el md5 seguirà la mateixa mecànica que el 1.1.4 per executar comandes.

1.2 Estructures de dades

En quant a la configuració del sistema:

1.2.1 Configurition

```
typedef struct configuration
{
    char* user;
    char* folder;
    char* IP;
    char* port;
    char* web;
    char* port_start;
    char* port_end;
} Configuration;
```

Figura 9: Estructura de configuració

Utilitzada per guardar la informació del fitxer de configuració.

En quant a les comunicacions, tenim les següents estrucutres de dades:

1.2.2 Connection

```
typedef struct connection //Struct for all the connections
{
    char* ID;
    int conn_fd;
    int port; /// Need for retrieving the name in show
connections
    char* path; //Extra! Need for this particular case
}Connection;
```

Figura 10: Estructura de connexió

Utilitzada tant per servidor com client.

En quant el servidor, utilitza els camps:

- **ID:** Cada thread del server necessita saber quin és el seu identificador per proporcionar als clients quan es connecten al seu port
- **Conn_fd:** Cada thread del server necessita tenir el file descriptor de la connexió amb el client que atén.
- **Path:** Cada thread del server necessita saber on es troben guardats els àudios per servir als clients.

En quant el client, utilitza els camps:

- **ID:** Cada client necessita saber el nom del server amb qui es connecta
- **Conn_fd:** Cada client necessita saber el file descriptor de la connexió que té amb el servidor
- **Port:** Cada client necessita saber el port corresponent a cada server per poder-lo relacionar amb el seu ID en la opció show connections

1.2.3 Llista amb punt d'interès dinàmica

Tant el client com el servidor, tenen múltiples servidors i clients als que connectar-se respectivament. Això requereix d'alguna estructura de dades on guardar les diverses connexions. Entre les estructures, s'ha triat una llista amb punt d'interès degut a la relació facilitat de programació – rendiment.

L'inserció i consulta de les dades és completament aleatòria, a raó de les peticions dels clients i servidors, per tant una llista de punt d'interès sembla una bona opció. Disposant de més temps es podria haver valorat d'usar una estructura més eficient, com les vistes a PAED.

L'element a guardar en aquesta llista serà una estructura de dades Connection.

A més a més, s'ha creat una llista amb punt d'interès abstracte, que permet guardar tot tipus d'estructures sense modificar la llista en sí.

```
typedef struct node
{
    void* e;
    struct node* next;
}Node;

typedef struct poiList
{
    Node* head;
    Node* prev;
}POIList;
```

Figura 11: Estructura de Llista amb Punt d'Interès

1.3 Recursos del sistema utilitzats

1.3.1 Signals

Es fa ús del signal SIGINT. En el moment en que un usuari apreta CONTROL + C, el signal crida a la rutina de neteja i tancat del procés.

També és fa ús del signal SIGPIPE per detectar servidors caiguts.

1.3.2 Pipes

La pràctica requereix de usar la línia de comandes dues vegades:

- Executar l'script show_connections
- Realitzar el md5 dels arxiu transferit

Això requereix usar la funció exec o algunes de les seves variants. El problema d'aquesta funció es que acaba amb el procés. Per aquesta raó, cal usar pipes en conjunció de forks.

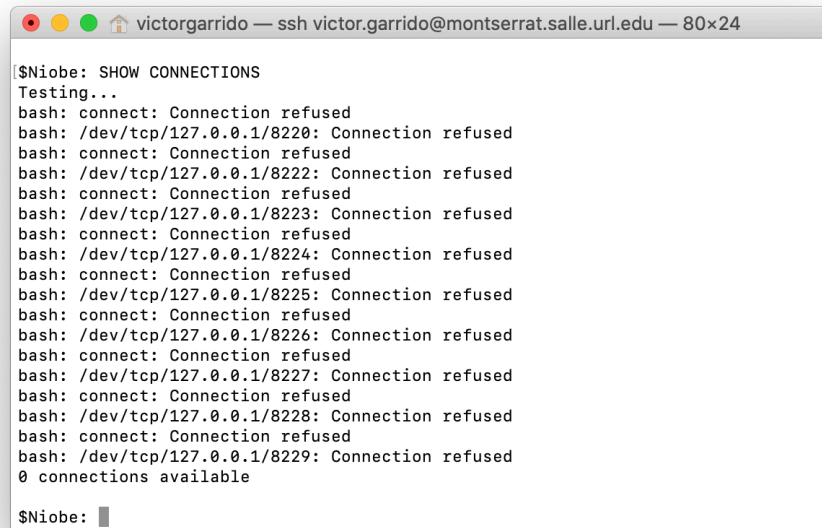
1.3.3 Exclusió mútua

El sistema requereix d'exclusió mútua simple en aquestes ocasions:

- S'escriu a la pantalla
- És vol accedir a la llista de clients connectats al servidor

2 Problemes Observats

En primer lloc, al executar `show_connections`, es mostren per pantalla outputs que teòricament estan silenciats. Això deu ser degut a algun problema del montserrat perquè no hauria d'aparèixer, donat que el script de per sé silencia els outputs de la comanda que executa



```
victorgarrido — ssh victor.garrido@montserrat.salle.url.edu — 80x24

$Niobe: SHOW CONNECTIONS
Testing...
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8220: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8222: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8223: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8224: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8225: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8226: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8227: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8228: Connection refused
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/8229: Connection refused
0 connections available

$Niobe: █
```

Figura 12: Problemàtica de show connections

En segon lloc, manejar memòria dinàmica ha portat moltíssims problemes, i més quan s'està treballant amb threads. Ha portat moltíssimes hores de feina subsanar tots els leaks. L'eina valgrind ha estat vital per tal de poder trobar i solucionar tots els leaks.

En tercer lloc, l'enviament del fitxers ha resultat problemàtic per el fet de que el client llegia més ràpid que rebia dades. Això portava una pèrdua de dades. S'ha solucionat obligant al client a llegir fins haver llegit tants bytes com s'havien enviat, és a dir, 500.

En quart lloc, l'arxiu de makefile ha resultat problemàtic i s'ha hagut de modificar per aconseguir que compilés el projecte a mesura que anava creixent en fitxers i directoris. El resultat és un makefile molt dinàmic i compatible amb molts tipus d'organització de projectes.

En cinquè lloc, fer ús del montserrat ha estat problemàtic per les limitacions que ofereix per alumne en quant als recursos. Hi havien moments en els quals m'havia d'esperar per tal de poder tornar a compilar.

3 Estimació temporal

Fases	DISSENY	IMPLEMENTACIÓ	TESTING
FASE 1	10	15	7
FASE 2	20	20	10
FASE 3	10	10	8
FASE 4	3	3	2

Afegir que la memòria m'ha portat 4 hores en realitzar

4 Conclusions i propostes de millora

Amb aquesta pràctica, he augmentat exponencialment el coneixement que tenia del llenguatge C així com la seva relació amb el sistema Linux. Del coneixement adquirit a 1r a el coneixement adquirit a 3r hi ha un abisme.

He pogut treballar amb el concepte dels files descriptors, tant com per llegir de teclat i escriure a pantalla, així com amb fitxers i amb connexions de xarxa.

He pogut treballar, i patit, extensament, d'utilització de la memòria dinàmica i com de difícil és d'implementar de manera correcta sense tenir cap leak. Això em fa dubtar de la seva escalabilitat i manteniment a nivell de codi i de si a vegades, seria més fructuós, encara que es perdessin uns quants bytes, utilitzar memòria estàtica.

He pogut treballar el concepte de client i servidor tot adonant-me de la complexitat que requereix per a què, quelcom tal senzill, que realitzem centenars de vegades amb els nostres dispositius al llarg del dia, funcioni de manera estable i assegurant que elements com la tramesa de fitxers funcioni com és d'esperar, sense l'ajuda d'un llenguatge d'alt nivell com Java. Un dels conceptes essencials per assegurar que la relació client-servidor funciona és el concepte de la concurrència.

He pogut treballar amb el concepte de la concurrència tot aplicant exclusió mútua per totes aquelles tasques que requereixen de zones crítiques.

He pogut treballar amb el concepte dels forks amb conjunció amb les pipes amb conjunció amb l'execució de comandes externes al procés tot processant la sortida.

Finalment, he pogut treballar amb el concepte dels threads tot aplicat a un servidor que atén múltiples peticions.

Puc valorar aquesta pràctica com molt completa i útil per el meu futur professional. Tots els elements vistos són utilitzats en un gran nombre de software comercial que utilitzem tots nosaltres diàriament. Acabo havent ampliat el meu coneixement exponencialment.

Com a propostes de millora m'hagués agradat tenir més temps per desenvolupar la pràctica de manera més modular i més abstracte per practicar més profundament el concepte de la reutilització del codi.

5 Bibliografia

- Pont, Jordi Salvador. 2009-2010. *Introducció al llenguatge de programació C*. 2009-2010.
- . 2014-2015. *Programació en UNIX*. 2014-2015.