LSBox

Autors:
Ferran Obiols Jordan
Xavier Canaleta LLampallas
Arturo Roversi Cahiz

Supervisors:
Marcos Herranz Ramírez
Oscar Salas Mestres
Aaron Jeriel Pica Pardos

7 de gener de 2013

Índex

1 Introducció conceptual a les pràctiques

Les pràctiques del Curs 2012/2013 de l'assignatura de Sistemes Operatius tenen com a objectiu que l'alumne aprengui a dissenyar i implementar diverses estratègies que un sistema operatiu té com a funcionalitats en els diferents mòduls dels quals es composa.

Un sistema operatiu està format, bàsicament, per 4 mòduls o subsistemes:

NUCLI. És l'única capa que pot inhibir interrupcions. Les seves funcions bàsiques són la representació dels processos en execució, el control de la concurrència de processos, la gestió i control de les interrupcions, i dotar al sistema de mecanismes de comunicació entre processos, de sincronització i exclusió mútua.

SISTEMA E/S. És l'única capa que pot executar instruccions del tipus entrada i sortida. La seva funció és la comunicació amb els perifèrics. Per tant, les capes superiors, per a dialogar amb el hardware, ho han fer a través d'aquesta capa.

GESTIÓ DE MEMÒRIA. Conjuntament amb el hardware, crea la possibilitat de disposar de memòria virtual. A més a més, aquesta capa és la responsable de garantir (també conjuntament amb el hardware) la protecció de les dades a memòria i també la seva compartició.

SISTEMA DE FITXERS. Dota al sistema d'una visió estructurada de les dades emmagatzemades al disc.

Les aplicacions d'usuari són programes creats per l'usuari. Quan aquests programes necessiten alguna funcionalitat del sistema operatiu, realitzen el que s'anomena una "crida al sistema", que consisteix en fer crides a funcions que ofereixen les diferents capes. És a dir, si l'aplicació vol mostrar un caràcter per pantalla, haurà de cridar a una funció de la capa d'E/S per a aconseguir-ho. Durant tot el curs s'aniran realitzant pràctiques de laboratori per tal que es puguin posar en pràctica els coneixements adquirits a les classes de teoria i complementar-los amb algunes innovacions pràctiques.

El contingut de la pràctica de l'assignatura de Sistemes Operatius està destinat a l'aprenentatge d'una arquitectura distribuïda mitjançant la utilització de sockets, posant de relleu tota la problemàtica de la concurrència de processos. Aquesta pràctica pretén que l'alumne adquireixi tots els mecanismes necessaris i conegui la problemàtica adjunta a un sistema distribuït. L'objectiu

principal és establir connectivitat entre els diferents equips i permetre la transmissió d'informació entre els diversos nodes d'un sistema distribuït. Addicionalment, cal la utilització de tots els mecanismes del nucli del sistema operatiu explicats al primer semestre (memòria compartida, mètodes d'exclusió mútua, sincronització i creació de processos), per a poder resoldre la pràctica.

NOTA 0. Les quatre pràctiques del curs 2004-2005 rebien els noms de Atreides, Atreides++, Harkonnen i Fremen. Tot era en honor a Frank Herbert i la seva famosa novel·la Dune. Força alumnes van recordar l'origen i van descobrir d'on provenia el nom.

NOTA 1. Les quatre pràctiques del curs 2005-2006 rebien els noms de Oedipus Rex, Sphinx, Antigona i Teiresias. Com totes les pràctiques de SO, no hi ha res a l'atzar. Els noms provenien de la tragèdia d'Èdip de Sòfocles i dotaven de sentit a les pràctiques. S'iniciava el curs amb una pràctica 1 senzilla, Èdip Rei. Èdip viu feliç com a rei mentre ignora que ha matat el seu pare i s'ha casat amb la seva mare. La ignorància el fa feliç. Els alumnes de SO viuen feliços perquè la pràctica 1 és fàcil i no saben que els espera. La pràctica 2 és l'Sphinx. Èdip ha de superar la prova de l'Sphinx ja que, sinó, aquesta el matarà. Òbviament, la pràctica 2 del curs 2005-2006 era la més difícil de totes i si no la superaves no aprovaves l'assignatura. La pràctica 3 es deia Antígona, qui ajuda a Èdip un cop aquest cau en desgràcia i en la desesperació. Antígona era la salvació dels alumnes que volien anar a l'examen de juny i necessitaven una pràctica llarga lliurada. Us estalvio els detalls de Teiresias. Tot i que molts alumnes van seguir els noms i les referències, cap d'ells va acabar de copsar el sentit de la tragèdia...

NOTA 2. Les pràctiques del curs 2006-2007 es van centrar en l'obra mestra de Dante Alighieri, "La Divina Commedia". Les pràctiques es deien "Inferno", "Purgatorio" i "Paradiso". Suposo que no calen gaire comentaris per entendre què passa a la pràctica 1. La segona pràctica es suavitza però la necessites si vols presentar-te a examen. Finalment, la tercera és relativament curta i senzilleta, un paradís comptant amb tot el que has fet anteriorment.

NOTA 3. Les pràctiques del curs 2007-2008 es van centrar en l'obra mestra dels germans Wachowski, "Matrix", on els humans (alumnes) han de lluitar contra les màquines per a sobreviure (aprovar l'assignatura). Les pràctiques es deien "Matrix", "El Ferroviario" i "Keymaker". Matrix era el servidor central on s'havien de connectar els clients, com Nebuchadnezzar, per a poder intercanviar arxius i comunicar-se mitjançant un xat distribuït. També podien connectar-se amb altres dimonis, com Oraculo o Link, que els donaven consells. El Ferroviario era una pràctica de simulació de càrrega de processos i administració de memòria. Si coneixeu el paper de El Ferroviario a la pel·lícula Matrix veureu que la relació és directa. Finalment, la codificació de fitxers en EXT2. Vam considerar que un bon nom per descodificar era el personatge de Matrix anomenat Keymaker.

NOTA 4. Les pràctiques del curs 2008-2009 es van centrar en l'obra mestra de Isaac Asimov, "La fundació". Les pràctiques es deien: "Trantor", el planeta central de la galàxia, tenint, per tant, la mateixa magnitud que la pràctica 1 de Sistemes Operatius; "Trevize, the loader", nom del conseller de la primera fundació, amb una intuïció insòlita, però amb perilloses intencions; i "Pelorat, the file reader", nom del professor d'història que ajudà a Trevize a trobar el planeta Terra, la llar on descansar després d'aprovar les tres pràctiques de Sistemes Operatius.

NOTA 5. Les pràctiques del curs 2009-2010 es van centrar en el grup australià de hard rock AC/DC, com a homenatge a la marxa del que va ser professor de pràctiques de l'assignatura els tres darrers anys, Hugo Meza. La primera pràctica, sota el nom de Highway to shell (petita modificació del nom de la cançó més popular del grup), conduïa als alumnes per tres fases. La primera, Welcome to the Shell, donava la benvinguda a l'infern als alumnes amb la implementació d'un intèrpret de comandes shell, sota el nom de Malcolm (cantant del grup). A la segona fase, per tal de fugir de les tenebres en què es trobaven, els alumnes es van veure pactant amb el diable (Dealing with the Devil), on havien de seguir tot un protocol per a comunicar-se amb el dimoni Angus (guitarrista del grup). Finalment, quan creien que tot havia passat, es topaven amb Ballbreaker (disc que van gravar al 1995), nom que no requereix gaires explicacions. La segona pràctica del curs 2009-2010 rebia el nom de Back in Black, nom del disc llençat al 1980, on els alumnes havien d'identificar el format d'un volum donat i extreure'n informació.

NOTA 6. Les pràctiques del curs 2010-11 es van centrar en les obres del polèmic Donatien Alphonse François de Sade, més conegut com "el Marqués de Sade". Així, la pràctica es titulava "Les 120 journées de Sodome". Aquesta estava dividida en tres fases: "Le Château de Silling" era la primera d'elles, on els alumnes gestionaven el comportament del client de l'aplicatiu, el qual rebia el nom del personatge de la novel·la "Blangis". Posteriorment els alumnes s'endinsaven al castell en una segona fase anomenada "Les quatre madames", on Blangis es connectava a un dimoni anomenat "Thérèse". Finalment, la cosa es desmadrava a la tercera fase, on els alumnes havien de crear el servidor "Libertinage", que, com el seu nom indica, és on començava el "festival".

NOTA 7. Les pràctiques del curs 2011-12 es van centrar en la mundialment coneguda serie televisiva del Simpsons. Aquesta pràctica estava dividida en cinc fases: "La shell de Homer", "el bar de Mou", "Clancy Wiggum" "Living in Springfield" i "Living in Springfield++", en el seu conjunt aquestes fases formaven un sistema que permetia executar comandes de forma local, algunes pròpies en un servidor remot i l'activació de diversos serveis que mostraven frases mítiques de la sèrie, tot això seguint una arquitectura client - servidor.

A partir d'aquí la interpretació dels noms i els conceptes de les pràctiques d'aquest any (relacionant la dificultat, el contingut i l'assignatura) us pertoca a vosaltres...

2 Descripció general de funcionalitats

La pràctica consistirà en implementar un sistema de compartició d'arxius online, **programat i funcionant a l'entorn UNIX de l'escola (servidors vela i/o cygnus)**. Aquest sistema estarà completament demonitzat, per lo que no hi haurà cap interacció directe amb l'usuari.

Fase 1 A la primera fase es crearan les aplicacions client i servidor. Aquestes dos aplicacions hauran de mantenir el contingut del directori arrel en memòria, implementat amb una estructura del tipus **Linked List**.

Fase 2 La segona fase s'establira la connexió entre el client i el servidor, realitzant la verificació de les credencials de l'usuari que s'està connectant.

Fase 3 La tercera fase s'haurà d'implementar un sistema de logs complert per tal de ser conscient del que està succeint en l'execució del nostre sistema.

Fase 4 La quarta fase és procedirà a la transmissió servidor - client.

Fase 5 En la cinquena fase es procedirà a realitzar la transmissió client - servidor - client, client, client

Fase 6 Una última fase per deixar el sistema preparat per demonitzar-lo i deixar-lo llest per producció.

Esquema final de la pràctica

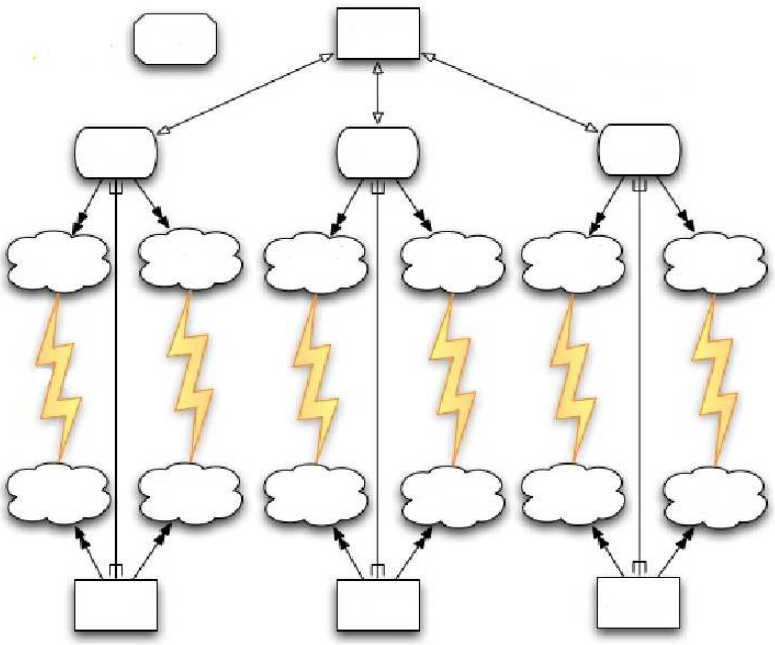

Figura 1: Esquema general de disseny

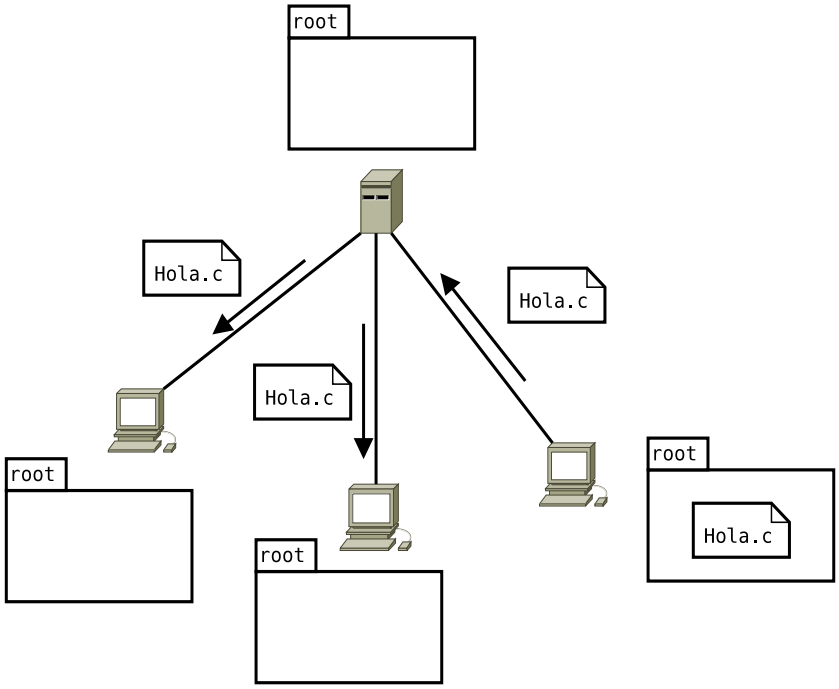

Figura 2: Esquema funcional exemple 1

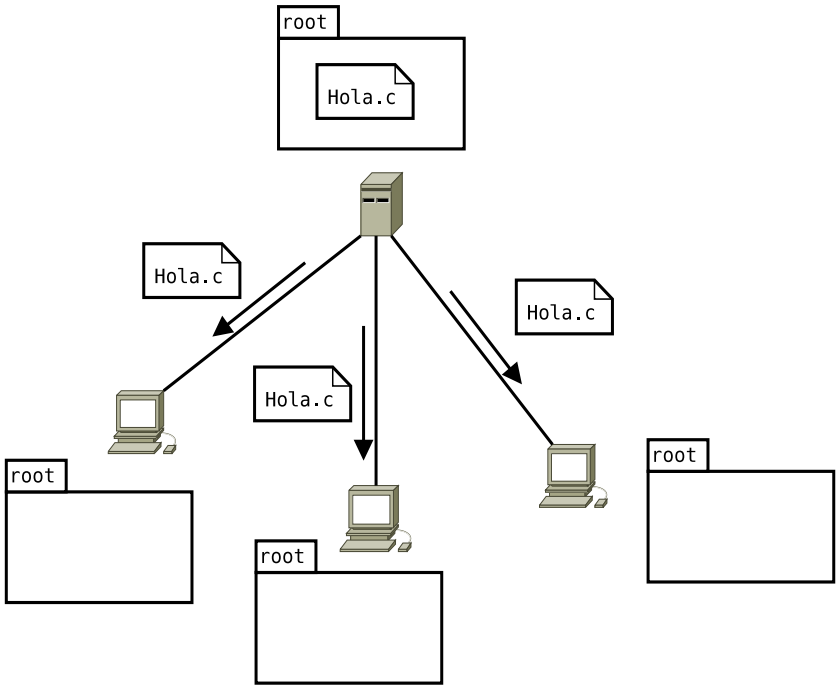

Figura 3: Esquema funcional exemple 2

3 Fase 1: Plantant Arrels

En aquesta primera fase de l'aplicació, simplement el que es pretén és fer una primera inicialització de tots els continguts que posteriorment seran necessaris en altres fases. Caldrà crear dos aplicacions, una serà l'aplicació client i una altra serà l'aplicació servidor.

Client

L'aplicació client en iniciar-se, demanarà a l'usuari que introdueixi login i contrasenya. En aquest moment, d'aquesta informació no se'n farà ús, però cal mantenir-la localitzada ja que serà necessària per la següent fase.

Un cop introduïda la informació, s'haurà d'inicialitzar un llista de directoris implementada amb una **Linked List** amb el contingut del directori root. Aquest directori root vindrà determinat al fitxer çonfig.dat. Un cop creada la llista s'haurà d'anar actualitzant/comprovant amb una cadència de **2 segons** per assegurar-se que la llista es manté al dia en tot moment.

La resta d'informació que conté l'arxiu de config.dat, en aquest moment no se'n farà ús, però també és recomanable tenir-la carregada i localitzada ja que en la següent fase serà necessària.

Servidor

L'aplicació servidor, simplement un cop iniciada crearà la **Linked List** del directori root del servidor i, igual que el client, haurà d'estar pendent que no s'afegeixi cap nou element directament al directori, però com que en aquest cas és menys probable que succeeixi, doncs no és habitual que s'afeigeixin coses directament al servidor, en aquest cas la cadència serà de **15 segons**.

Linked List

La llista que s'haurà d'implementar, haurà de ser un TAD declarat en un arxiu a part del codi i el qual sigui fàcil de reutilitzar. El disseny i la implementació d'aquest TAD és decisió de l'alumne.

Patró de l'arxiu config.dat:

```
1 <Server>
2 <Port>
3 <Root diretori Path>
```

Observacions:

- NO es poden utilitzar les funcions printf, scanf, gets, puts, etc. Només es podrà interactuar amb la pantalla amb les funcions read (lectura) i write (escriptura). Sí que es permet fer ús de la funció sprintf.
- NO es poden utilitzar les funcions system, popen ni cap variant. La creació de processos només es podrà fer utilitzant forks o threads.
- Cal garantir l'estabilitat de l'aplicació i el seu correcte funcionament: en cap cas es poden produir bucle infinits, core dumpeds, etc. També cal controlar tots aquells aspectes susceptibles de donar algun error i, en cas que es produeixin, informar degudament a l'usuari i, si és possible, seguir amb el funcionament normal de l'aplicació.
- Des d'ara i fins al final de la pràctica cal considerar que qualsevol procés pot morir inesperadament i sense notificar-ho (per exemple rebent un kill -9). Cal tenir-ho en compte i procurar perquè tot el sistema quedi el més estable possible en cas que es produeixi.

4 Fase 2: Creant Llaços

En aquesta segona fase, es procedirà a establir connexió entre el client (LSBox_cli) i el servidor (LSBox_svr):

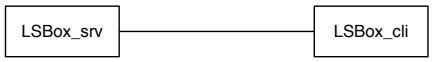

Figura 4: Connexió entre els dos programes

Aquesta connexió es crearà a través d'un socket i, un cop establerta, el client es comunicarà amb el servidor per enviar-li les credencials que tenim guardades de la Fase 1, el protocol de comunicació a utilitzar es troba a l'Annex. Com es pot observar la password no s'enviarà tal qual, sinó que s'enviarà la cadena **md5** del password.

La comprovació de la contrasenya la farà el procés pare del servidor. El pare el que farà és comparar la contrasenya que li arriba amb la que té ell emmagatzemada dins del fitxer shadow.dat i respondre en conseqüència. Si el registre ha resultat satisfactori s'haurà de crear una connexió dedicada i el fill (LSBox_child) haurà de mantenir el login de l'usuari, doncs posteriorment serà necessari per fer validacions de trames.

S'ha de tenir en compte que s'ha de validar la informació de totes i cada una de les trames. Això significa que s'ha de comprovar sempre que el login origen i destí siguin els correctes i possiblement en alguns casos que el tipus de la trama també sigui la que toca.

L'arquitectura que haurà de quedar al final d'aquesta fase seria:

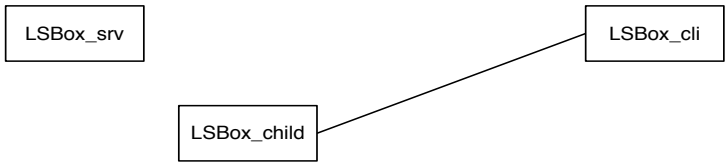

Figura 5: Connexió establerta

Shadow.dat:

Aquest és el fitxer que consulta el servidor per saber els usuaris que poden accedir i les seves contrasenyes emmagatzemades amb md5. Aquest fitxer es crearà **manualment**.

```
1 <login1>:<md5 password1>
2 <login2>:<md5 password2>
3 <login3>:<md5 password3>
4 <login4>:<md5 password4>
```

Observacions:

- És obligatori la utilització d'un makefile per generar els dos executables.
- NO es poden utilitzar les funcions printf, scanf, gets, puts, etc. Només es podrà interactuar amb la pantalla amb les funcions read (lectura) i write (escriptura). Sí que es permet fer ús de la funció sprintf.
- NO es poden utilitzar les funcions system, popen ni cap variant. La creació de processos només es podrà fer utilitzant forks o threads.
- Cal garantir l'estabilitat de l'aplicació i el seu correcte funcionament: en cap cas es poden produir bucle infinits, core dumpeds, etc. També cal controlar tots aquells aspectes susceptibles de donar algun error i, en cas que es produeixin, informar degudament a l'usuari i, si és possible, seguir amb el funcionament normal de l'aplicació.
- Des d'ara i fins al final de la pràctica cal considerar que qualsevol procés pot morir inesperadament i sense notificar-ho (per exemple rebent un kill -9). Cal tenir-ho en compte i procurar perquè tot el sistema quedi el més estable possible en cas que es produeixi.

5 Fase 3: Escrivint al Diari

Abans de procedir amb coses més complicades, és imprescindible que implementeu un sistema de logs. Un log és una entrada de text en un fitxer anomenat arxiu de logs. Aquestes entrades ens permeten veure que és el que ha anat malament, que és el que ha anat bé o simplement que és el que s'ha fet. Per tant, el que haureu de fer és, que el LSBox_svr, LSBox_cli i LSBox_child creïn uns fitxers (si no està creat) amb el noms següents:

- LSBox_svr = LSBox_svr.log
- LSBox_child = <login_usuari>.log
- LSBox_cli = LSBox_cli.log

Cada un d'aquest arxius haurà de tenir com a mínim els següents logs:

- Trames Rebudes.
- Trames Enviades.
- Comprovacions de modificacions en el directori root.
- Modificacions en el contingut de la **Linked List** del directori root.
- Errors que es puguin produir durant l'execució.
- Creació de Threads o Forks.
- Llistat d'elements que es sincronitzaran a través del Thread.
- Bloquejos de memòria.
- Creació o eliminacions de memòria compartida.
- Duplicacions de file descriptors.

Nomenclatura

El format que es seguirà per les entrades de log serà:

[arxiu][funció]explicació[KO/OK]

On el diferents camps signifiquen:

- **Arxiu** Nom de l'arxiu .c que conté la funció.
- **funció** Nom de la funció que rep, envia, executa, comprova alguna funcionalitat/trama que provoca un resultat correcte o incorrecte en algun dels casos mencionats anteriorment.
- **Explicació** Explicació de l'error, la trama rebuda, la trama enviada, la comprovació, o l'execució que s'ha realitzat.
- **KO/OK** resultat obtingut amb la funció OK correcte, KO incorrecte.

Format

Aquests logs hauran de seguir un format de html per tal de poder-ne fer una lectura senzilla a traves d'un navegador web. El format d'un document de html és:

```
1
2 <!DOCTYPE html>
3 <html lang="en">
4
5   <head>
6     <meta charset="utf-8" />
7     <title>[SO] Logs</title>
8
9   </head>
10
11  <body id="index" class="home">
12
13  </body>
14 </html>
```

I el format de les entrades de log, que van entre les etiquetes de **body**, seguint l'esquema anterior seran:

```
1 <p>
2   <font color="blue">[</font> <font color="teal"> ... </font><font
      color="blue">]</font>
3   <font color="blue">[</font> <font color="red"> ... </font><font
      color="blue">]</font>
4   <font color="black"> ... /font>
5   <font color="blue">[</font> <font color="green/red"> ...
      </font><font color="blue">]</font>
6 </p>
```

En l'últim cas, com es pot observar, dins del font color hi ha green/red, en cas de que el resultat sigui OK (correcte) el contingut serà green i en cas de que sigui KO (incorrecte), el contingut serà red.

Observacions:

- És obligatori la utilització d'un makefile per generar els dos executables.
- NO es poden utilitzar les funcions printf, scanf, gets, puts, etc. Només es podrà interactuar amb la pantalla amb les funcions read (lectura) i write (escriptura). Sí que es permet fer ús de la funció sprintf.
- NO es poden utilitzar les funcions system, popen ni cap variant. La creació de processos només es podrà fer utilitzant forks o threads.

- Cal garantir l'estabilitat de l'aplicació i el seu correcte funcionament: en cap cas es poden produir bucle infinits, core dumpeds, etc. També cal controlar tots aquells aspectes susceptibles de donar algun error i, en cas que es produeixin, informar degudament a l'usuari i, si és possible, seguir amb el funcionament normal de l'aplicació.
- Des d'ara i fins al final de la pràctica cal considerar que qualsevol procés pot morir inesperadament i sense notificar-ho (per exemple rebent un kill -9). Cal tenir-ho en compte i procurar perquè tot el sistema quedi el més estable possible en cas que es produeixi.

6 Fase 4: Enviant senyals

Ara un cop tenim el terreny preparat, ja podem començar amb la part complicada del nostre sistema. En aquesta fase s'haurà de crear el sistema de transmissió d'arxius entre el servidor i els clients, i s'haurà de garantir la integritat tant dels arxius com de la **Linked List** del contingut del directori root.

Per fer-ho fàcil i evitar complicacions, començarem per un enviament unidireccional de la informació del servidor cap al client. Per tant, el procediment serà:

1. Posarem un arxiu a la carpeta del servidor.
2. El servidor actualitzarà la llista.
3. El servidor es comunicarà amb el client per saber quins són els arxius a sincronitzar.
4. El servidor (si té arxius a sincronitzar), crearà un thread per l'enviament, i enviarà una trama indicant-l'hi al client el port que s'ha de connectar.
5. El client es connectarà en el socket del thread.
6. El thread enviarà els arxius necessaris.
7. Quan el thread acabi aquest acabarà.

Concretem una mica més

Com ja s'ha dit en fases anteriors, el servidor comprovarà el contingut del directori root de LS-Box_svr amb una cadència de **15 segons**. Quan detecti una modificació respecte el que tenia inicialment, tant si es tracta d'una modificació d'un arxiu com la aparició d'un nou arxiu, el servidor haurà de notificar als seus fills que s'ha actualitzat la llista i, per tant, cada un d'ells procedirà a demanar als clients que li enviïn la informació de la seva llista.

El client enviarà la informació sobre el seu directori root amb les trames especificades a l'Annex, com es pot veure, s'enviarà una trama per cada arxiu tingui el client. Quan s'hagin enviat totes les trames i el servidor sàpiga quins són el arxius que s'han d'enviar o s'han d'esborrar, aquest crearà un thread paral·lel al fill que serà l'encarregat d'enviar la informació i alhora el client es connectarà a través d'un thread al thread del servidor a través d'un socket en el port indicat per la trama P.

Un cop finalitzada la sincronització i enviada la trama de finalització, el thread servidor tancarà el socket i tant el thread del servidor com el del client finalitzaran.

Observacions:

- És obligatori la utilització d'un makefile per generar els dos executables.
- Per tal de garantir la integritat del sistema de trames es garanteix que mai el nom d'un document superarà els 78 caràcters.

- **Quan es realitza la sincronització no serà vàlid que es descarreguin tots els arxius independentment que hagin estat modificats o no i, per tant, no es faci un bon ús dels recursos del sistema**
- NO es poden utilitzar les funcions printf, scanf, gets, puts, etc. Només es podrà interactuar amb la pantalla amb les funcions read (lectura) i write (escriptura). Sí que es permet fer ús de la funció sprintf.
- NO es poden utilitzar les funcions system, popen ni cap variant. La creació de processos només es podrà fer utilitzant forks o threads.
- Cal garantir l'estabilitat de l'aplicació i el seu correcte funcionament: en cap cas es poden produir bucle infinits, core dumpeds, etc. També cal controlar tots aquells aspectes susceptibles de donar algun error i, en cas que es produeixin, informar degudament a l'usuari i, si és possible, seguir amb el funcionament normal de l'aplicació.
- Des d'ara i fins al final de la pràctica cal considerar que qualsevol procés pot morir inesperadament i sense notificar-ho (per exemple rebent un kill -9). Cal tenir-ho en compte i procurar perquè tot el sistema quedi el més estable possible en cas que es produeixi.

7 Fase 5: Compartint Senyals

Ara que ja tenim arxius sincronitzant-se en una direcció, caldrà que es sincronitzi també en l'altra. Però aquest operació és una mica més delicada de realitzar ja que **caldrà garantir la integritat de la informació** de la **Linked List** i dels arxius.

El procediment a seguir és molt similar al que ja s'ha realitzat en la fase anterior. El Client, quan detecti que s'ha produït una modificació en el contingut de la **Linked List** del contingut del directori root, farà un petició de sincronització en el servidor i aquest acceptarà la sincronització i procedirà a fer el mateix procediment que en la fase anterior. Però com ja s'ha dit, és molt important garantir la integritat de la informació, tant a nivell de programa com a nivell físic. Per tant s'haurà de dotar a l'aplicació de mecanismes que ens permetin garantir-ho.

Un cop garantida la integritat de la informació caldrà muntar un sistema de broadcast per tal de que tots els clients es mantinguin sincronitzats en tot moment.

Observacions:

- És obligatori la utilització d'un makefile per generar els dos executables.
- **Quan es realitza la sincronització no serà vàlida l'opció de descarregar tots els arxius independement que hagin estat modificats o no i, per tant, no es faci un bon ús dels recursos del sistema**
- NO es poden utilitzar les funcions printf, scanf, gets, puts, etc. Només es podrà interactuar amb la pantalla amb les funcions read (lectura) i write (escriptura). Sí que es permet fer ús de la funció sprintf.
- NO es poden utilitzar les funcions system, popen ni cap variant. La creació de processos només es podrà fer utilitzant forks o threads.
- Cal garantir l'estabilitat de l'aplicació i el seu correcte funcionament: en cap cas es poden produir bucle infinits, core dumpeds, etc. També cal controlar tots aquells aspectes susceptibles de donar algun error i, en cas que es produeixin, informar degudament a l'usuari i, si és possible, seguir amb el funcionament normal de l'aplicació.
- Des d'ara i fins al final de la pràctica cal considerar que qualsevol procés pot morir inesperadament i sense notificar-ho (per exemple rebent un kill -9). Cal tenir-ho en compte i procurar perquè tot el sistema quedi el més estable possible en cas que es produeixi.

8 Fase 6: Espantant dimonis

En aquesta última fase, s'hauran de demonitzar tant el client com el servidor. Per tal de fer-ho correctament, s'haurà de refactoritzar part del codi que ja havíem implementat. En el cas del client, un cop logejat satisfactòriament, es demonitzarà i passarà a estar en segon pla. I en el cas del servidor aquest es demonitzarà directament només començar.

Observacions:

- És obligatori la utilització d'un makefile per generar els dos executables.
- NO es poden utilitzar les funcions printf, scanf, gets, puts, etc. Només es podrà interactuar amb la pantalla amb les funcions read (lectura) i write (escriptura). Sí que es permet fer ús de la funció sprintf.
- NO es poden utilitzar les funcions system, popen ni cap variant. La creació de processos només es podrà fer utilitzant fork's o threads.
- Cal garantir l'estabilitat de l'aplicació i el seu correcte funcionament: en cap cas es poden produir bucle infinits, core dumpeds, etc. També cal controlar tots aquells aspectes susceptibles de donar algun error i, en cas que es produeixin, informar degudament a l'usuari i, si és possible, seguir amb el funcionament normal de l'aplicació.
- Des d'ara i fins al final de la pràctica cal considerar que qualsevol procés pot morir inesperadament i sense notificar-ho (per exemple rebent un kill -9). Cal tenir-ho en compte i procurar perquè tot el sistema quedi el més estable possible en cas que es produeixi.

9 Requeriments de lliurament i planificació

Aquesta pràctica disposarà de diferents punts de control o checkpoints per poder fer-ne un seguiment acurat i garantir la consolidació de cadascuna de les fases de manera incremental. Concretament se seguirà el següent calendari de dates límit:

CONCEPTE	DEADLINE
Checkpoint Fase 1	21/10/2012
Checkpoint Fase 2	30/10/2012
Checkpoint Fase 3	15/11/2012
Checkpoint Fase 4	15/12/2012
Final Fase 1..5	Gener
Final Fase 1..6	Febrer

Els checkpoints no són obligatoris, tot i ser altament recomanables per poder garantir la robustesa de la pràctica. Aquests checkpoints només serviran per incrementar linealment la qualificació de la pràctica. En cap cas penalitzaran la seva nota.

També és recomanable validar el disseny global de la pràctica amb els monitors de pràctiques abans d'iniciar les fases 4 i 5. Així podreu garantir que la implementació no patirà d'inconsistències insalvables per fases posteriors. Per això només cal aprofitar els horaris de dubtes i validar els vostres dissenys.

Entrega final

Els requeriments mínims per aprovar aquesta pràctica dependran del moment en que es lliura, així com la nota màxima a la que es pot optar. A la següent taula es pot observar la nota màxima a la que s'aspira segons el nombre de fases entregades i el moment en que s'entreguen.

		Convocatòria			
		Gener	Febrer	Maig	Setembre
Fases	1..5	10	9	7	5
	1..6	10*	10	8	6

A cada lliurement final (Gener, Febrer, Maig, Setembre) es permetran tres "presentacions". Aixo no significa que un cop hagi finalitzat el termini límit d'entrega i una pràctica sigui suspesa es podrà re-entregar fins a dos cops més, sinó que significa, que fins la data del lliurement es podran fer tres lliuraments, en el moment del tercer lliurement aquest ja comptarà com a entrega definitiva d'aquella convocatòria. En aquetes entregues, els monitor faran una pre-comprovació de les funcionalitats del codi i, si la pràctica compleix amb les funcionalitats, els monitors donaran el lliurament per acceptat o en cas contrari s'informarà dels errors als alumnes per a que puguin fer un altra lliurament.

En cas de que la pràctica sigui acceptada, no significa que els integrants dels grups estiguin aprovats, ja que per poder aprovar la pràctica **s'ha de passar una entrevista** en la que els monitors faran preguntes teòriques tant de la pràctica que pròpiament han implementat els alumnes

com preguntes teòriques de conceptes i eines que cal conèixer per haver desenvolupat la mateixa. Si es constata que els coneixements són insuficients, s'aplicacran els punts de la normativa pertinents.

Els lliuraments es realitzaran a l'eStudy en un pou amb un fitxer que inclogui el codi font (fitxers .c, .h) i **un únic makefile** de la pràctica, funcionant completament sobre Vela. El fitxer haurà de ser obligatòriament en format .tar. El podeu generar amb la comanda següent:

tar cf Gx_Fn_login1_login2.tar *.c *.h makefile

On Fn és el número de Fase a lliurar. Per exemple, si el grup 12 està format pels logins st13886 i st13935 llavors seria G12_F1_st13886_st13935.tar per al lliurament del checkpoint amb la Fase 1. Una pràctica amb format d'entrega incorrecte no serà apta.

En els lliuraments finals també caldrà dipositar la memòria en format PDF. La memòria ha de constar, obligatòriament, dels punts que s'indiquen més endavant.

Observacions importants No és suficient que l'arxiu que pugeu al pou tingui extensió .tar, sinó que s'ha de poder desempaquetar amb la comanda tar. Qualsevol pràctica o checkpoint que no es pugui "descomprimir" d'aquesta manera **no serà corregida**.

Recordeu que el codi ha d'estar degudament modulat, és a dir: no es pot ubicar tot en un sol fitxer .c, i és obligatori que hi hagi un fitxer makefile. Tingueu en compte que si en intentar corregir una pràctica aquesta no compila amb la comanda make (pel motiu que sigui), l'entrega serà qualificada com a no apta.

Memòria
Ha de contenir exactament els següents apartats:

1. Portada
2. Índex
3. Per cada fase cal:
   a) Requeriments: què s'ha de fer en aquesta fase
   b) Disseny: explicació de com heu dissenyat i estructurat la fase
   c) Diagrames: processos, comunicació...
   d) Com s'han assolit els requeriments
   e) Estructures de dades usades (piles, cues, arrays...)
   f) Explicació dels recursos emprats (signals, semàfors...)
   g) Proves realitzades (testing)
   h) Problemes observats i com s'han solucionat
   i) Estimació temporal
   j) Conclusions i comentaris
4. Conclusions i comentaris generals

Referències

[1] KERRISK, M. (2010). The Linux Programming Interface: A Linux and UNIX System Programming Handbook. No Starch Press.

[2] RIFFLET, J.M. (1998). Comunicaciones en UNIX. McGraw Hill.

[3] STEVENS, W. R. (1999). UNIX Network Programming, Volume 2, Second Edition: Interprocess Communications. Prentice Hall, ISBN 0-13-081081-9.

[4] STEVENS, W. R., FENNER, B., RUDOFF, A.M. (2004). UNIX Network Programming, Volume 1: The sockets networking API. Ed. Addison-Wesley Professional.

[5] STEVENS, W. R., RAGO, S.A. (2008). Advanced Programming in the UNIX Environment, Second Edition. Addison-Wesley, ISBN 0-321-52594-9

[6] SALVADOR, J. (2012). Programació en UNIX per a pràctiques de Sistemes Operatius, Enginyeria i Arquitectura La Salle.

10 Annex I: Definició del protocol de comunicació

Per dur a terme la comunicació entre el procés i el dimoni i, posteriorment, entre el procés i qualsevol dels processos-servei, s'utilitzarà un protocol específic d'enviament de trames que s'explicarà a continuació. S'ha de tenir en compte que els missatges s'enviaran mitjançant sockets utilitzant una comunicació orientada a connexió.

En aquest protocol s'utilitzarà un únic tipus de trama. D'aquesta manera es simplifica el protocol i totes les trames que s'envien tenen exactament la mateixa mida. Aquesta tindrà 3 camps ocupant sempre un espai total de 115 bytes :

Login origen: array de 7 caràcters que contindrà el login de qui envia la trama. Si el login no arriba a 7 caràcters, la resta de bytes s'han d'omplir amb '\0'. Aquest camp sempre ocupa exactament 7 bytes.

Login destí: array de 7 caràcters que contindrà el login a qui va dirigida la trama. Si el login no arriba a 7 caràcters, la resta de bytes s'han d'omplir amb '\0'. Aquest camp sempre ocupa exactament 7 bytes.

Tipus: indica el tipus de trama que s'està enviant. Depenent de la trama aquesta tindrà diferents valors tal i com s'especifica posteriorment. Aquest camp ocupa sempre només 1 byte.

Data: camp de 100 bytes. Aquest camp serveix per emmagatzemar-hi valors o dades que ha d'enviar la trama. Depenent del cas pot contenir diferents valors com s'especifica en cada funcionalitat.

És molt important de cara a garantir la bona comunicació amb el dimoni que la definició de l'estructura de la trama de comunicació sigui estrictament, en ordre i format, la descrita amb anterioritat. Això vol dir que qualsevol canvi de tipus dels camps o ordre dels mateixos dins de l'estructura a dissenyar farà que les trames rebudes o enviades al dimoni no compleixin el protocol de comunicació i, per tant, no funcioni aquesta comunicació correctament.

11 Annex II: Protocol de comunicació Client - Servidor

11.1 Establiment de connexió

Quan s'estableix la connexió, el servidor envia una trama sol·licitant que el client s'autentifiqui. Aquest trama tindrà els següents camps:

- El camp login origen posarà "LSBox".
- El camp login destí posarà "client".
- El camp tipus posarà una 'P' de petició.
- El camp data contindrà el missatge: "petició d'autentificació"

El client pot contestar de varies formes:

1. Si el client rep una trama amb format incorrecte:

   Si el client rep la trama de forma incorrecta informarà al servidor utilitzant una trama amb la següent informació i passarà a estar a l'espera de altres peticions.

   - El camp login origen posarà el que ha facilitat l'usuari.
   - El camp login destí posarà "LSBox".
   - El camp tipus posarà una 'E' de petició.
   - El camp data contindrà el missatge: "Error en la petició d'autentificació".

2. Si el client rep la petició correctament:

   Si el client rep la petició correctament, enviarà el login i contrasenya que havíem demanat a la Fase 1, però en el cas de la contrasenya anirà encriptada amb md5 per tal de garantir la seguretat de la nostra informació.

   - El camp login origen posarà el que ha facilitat l'usuari.
   - El camp login destí posarà "LSBox".
   - El camp tipus posarà una 'A' de petició.
   - El camp data contindrà el loguin i contrasenya de l'usuari seguint el següent format: "<login>:<md5Contrasenya>".

El servidor es comportarà de la següent forma depenent de la trama que li arribi.

1. Si el servidor rep una trama incorrecte, una trama d'error o amb credencials incorrectes.

   Si el servidor rep alguna d'aquestes dos casuístiques, el servidor respondrà amb la següent trama

   - El camp login origen posarà "LSBox".

- El camp login destí posarà el que ha facilitat l'usuari.
- El camp tipus posarà una 'E' de petició.
- El camp data contindrà el missatge: "Error en el procediment d'autentificació. Procedim a la desconnexió.".

En rebre aquesta trama, el client tancarà el socket i el servidor, en detectar que el socket s'ha tancat, haurà de finalitzar el proces dedicat.

2. Si el servidor rep una trama correcte amb credencials correctes.

   En aquest cas es respondrà amb una trama de tipus O indicant que la connexió ha estat establerta satisfactòriament, i es passara a esperar altres tipus de trames.

   - El camp login origen posarà "LSBox".
   - El camp login destí posarà el que ha facilitat l'usuari.
   - El camp tipus posarà una 'O' de petició.
   - El camp data contindrà el missatge: "Autentificació realitzada satisfactòriament".

11.2 Sincronització

11.2.1 Petició procedent del client

Si el client és el qui vol iniciar una sincronització, enviarà la següent trama al servidor:

- El camp login destí posarà el que ha facilitat l'usuari.
- El camp login origen posarà "LSBox".
- El camp tipus posarà una 'L' de petició de sincronització.
- El camp data contindrà el missatge: "Petició de Sincronització".

Un cop rebuda aquesta trama es procedira amb el mateix procediment que es faria si fos els servidor qui inicies la sincronització.

11.2.2 Peticio Procedent del Servidor

Quan es vulgui procedir a la sincronització (que no és res mes que la comparació de la llista del client amb la llista del servidor) s'iniciarà amb la següent trama enviada des del servidor.

- El camp login origen posarà "LSBox".
- El camp login destí posarà el que ha facilitat l'usuari.
- El camp tipus posarà una 'S' de petició de sincronització.
- El camp data contindrà el missatge: "Inici sincronització".

Com a resposta a aquest missatge, el client pot respondre amb un missatge d'error o un missatge de tipus O:

1. Si el servidor rep una trama amb format incorrecte:

   - El camp login origen posarà el que ha facilitat l'usuari.
   - El camp login destí posarà "LSBox".
   - El camp tipus posarà una 'E' de petició.
   - El camp data contindrà el missatge: "Error de trama".

2. Si el client rep la petició correctament:

   Si el client rep la petició correctament, enviarà una trama de confirmació.

   - El camp login origen posarà el que ha facilitat l'usuari.
   - El camp login destí posarà "LSBox".
   - El camp tipus posarà una 'O' de petició.
   - El camp data contindrà el missatge: "Sincronització confirmada".

Un cop es confirma la sincronització el client procedirà a enviar un per un la informació dels arxius sense esperar confirmació alguna. El format de les trames que enviarà és:

- El camp login origen posarà el que ha facilitat l'usuari.
- El camp login destí posarà "LSBox".
- El camp tipus posarà una 'N' de petició.
- El camp data contindrà el missatge: "<nom>:<data darrera modificació>:<mida>".

Quan ja no quedin més arxius a enviar, s'enviarà una de tipus X:

- El camp login origen posarà el que ha facilitat l'usuari.
- El camp login destí posarà "LSBox".
- El camp tipus posarà una 'X' de petició.
- El camp data contindrà el missatge: "Sincronització finalitzada".

11.3 Enviament de arxius

En aquest moment, el servidor ja té la informació necessari per començar la retransmissió, per tant el primer que farà és crear el thread i començar a escoltar en un port lliure. Aquest port se li enviarà al client perquè crei un thread que es connecti al thread servidor.

- El camp login origen posarà "LSBox".
- El camp login destí posarà el que ha facilitat l'usuari.
- El camp tipus posarà una 'P' de petició.

- El camp data contindrà el missatge: "<PORT>".

Un cop connectat procedirà a l'enviament de les trames per tal de que a les dos bandes tinguin la mateixa informació. Hi ha dos tipus de trames:

1. Si s'afegeix/Modifica document:

   - El camp login origen posarà "LSBox".
   - El camp login destí posarà el que ha facilitat l'usuari.
   - El camp tipus posarà una 'M' de creació.
   - El camp data contindrà el missatge: "<nom de l'arxiu>:<mida de l'arxiu>:<data darrera modificacio>".

2. Eliminació d'un arxiu:

   - El camp login origen posarà "LSBox".
   - El camp login destí posarà el que ha facilitat l'usuari.
   - El camp tipus posarà una 'D' de delete.
   - El camp data contindrà el missatge: "<nom de l'arxiu>".

En el cas que s'hagi enviat una trama de tipus 'M', la informació del contingut s'enviarà amb la següent trama.

- El camp login origen posarà "LSBox".
- El camp login destí posarà el que ha facilitat l'usuari.
- El camp tipus posarà una 'T' de transmissió.
- El camp data contindrà la informació del document (100 bytes).

Per últim, per indicar des del servidor que ha acabat d'enviar les trames referents als arxius, enviarem una trama de tipus X. Moment en el qual el client procedirà a tancar la connexió del socket.

- El camp login origen posarà "LSBox".
- El camp login destí posarà el que ha facilitat l'usuari.
- El camp tipus posarà una 'X' de petició.
- El camp data contindrà el missatge: "Enviament finalitzat".

12 Exemples

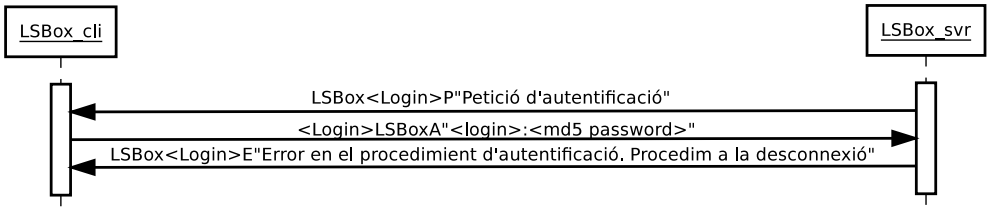

Figura 6: Error en l'autentificació

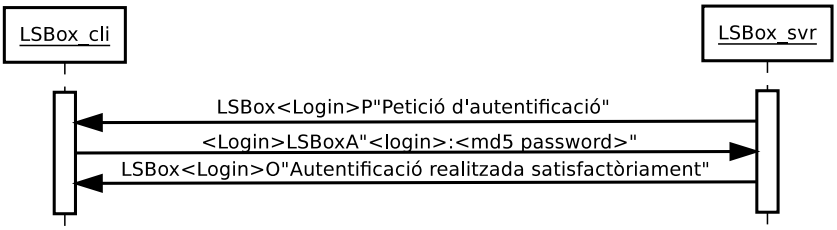

Figura 7: Autentificació Correcta

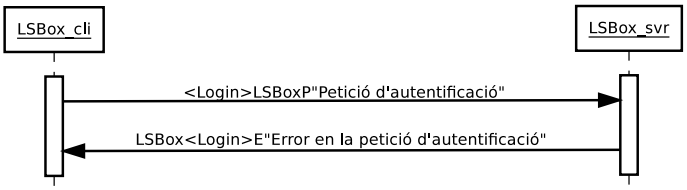

Figura 8: Error en la recepció d'una trama

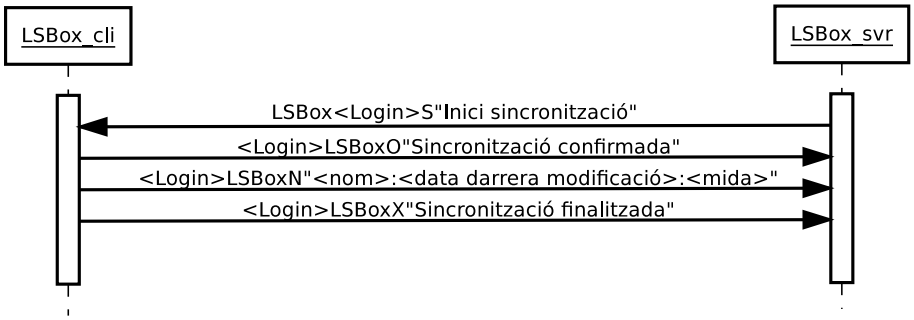

Figura 9: Sincronització de la llista