GESTIÓ I DESENVOLUPAMENT D'UN CONJUNT COMPRENSIU DE PRÀCTIQUES PER LES ASSIGNATURES DE SISTEMES OPERATIUS

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Joan Caparrós Ramírez
i dirigit per
Eduardo Cesar Galobardes
Bellaterra, de setembre de 2007

"Qui estudia i no posa en practica la ciència, és un llaurador que llaura i no sembra."

Saad

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Eduardo Cesar Galobardes

Professor de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Joan Caparrós Ramírez

I per tal que consti firma la present.

Signat: .....

Bellaterra, de setembre de 2007

Agraïments

En primer lloc vull agrair a l'Eduardo Cesar la gran implicació que ha demostrat, així com el suport i els consells que m'ha ofert.

Als companys d'estudi (Alejandro B., J. M. Elias, Toni, Pau, Mayte, etc) que any darrera any ens em anat donant suport.

Agraeixo sobretot el suport del meu pare i el meu germà.

I a tots als professors que han treballat durament per a moldejar la nostre ment. Gràcies a tots.

Índex

1. INTRODUCCIÓ

L'**objectiu principal del projecte** és estudiar i implementar un repositori de pràctiques per a les assignatures de Sistemes Operatius de la Universitat Autònoma de Barcelona, intentant innovar i aprofundir en aspectes no tractats fins ara, però essencials per a un bon aprofundiment en el material impartit en les classes de teoria.

El projecte conta amb dues vies d'estudi i desenvolupament, per una part el propi anàlisis del conjunt de pràctiques i per un altre la incorporació d'una nova eina per el control i administració del conjunt de pràctiques de les que es disposa.

Respecte la primera basant del àrea d'estudi remarcarem la idea original del **Pla Pilot d'adaptació a Bolonya,** un sistema on es redueixen les hores de docència directa (classes magistrals) i es dóna més importància a tot el procés d'aprenentatge de l'estudiant, que tindrà un rol més actiu i participatiu durant tot el seu procés de formació. L'estudiant aprendrà d'una manera diferent i amb una avaluació que va més enllà dels clàssics exàmens, i que tindrà en compte el seu esforç (treballs, estudi, seminaris, treballs en equip, tutories, etc.) [UAB]. Les sessions pràctiques realitzades fins al moment tractaven sobre pocs aspectes en els quals s'aprofundia molt, el nou pla d'estudis, donant més importància a l'aprenentatge del estudiant, ens permetrà un conjunt més difós d'estudi en els que les anteriors pràctiques no seran excloses però que podran ser complementades amb nous materials, que ampliaran els temes practicats fins al moment.

El segon enfoc d'aquest projecte recau en la organització i control de les pràctiques existents, desenvolupant un portal web on tots els membres del personal docent podran consultar i introduir material per al desenvolupament de les pròpies pràctiques, deixant així un conjunt ordenat que ajudi als professors a ensenyar i impartir aquestes classes, basant-se en aportacions d'altres companys que han realitzat aquestes tasques amb anterioritat (degudes a rotacions del personal).

Per tal d'elaborar un conjunt de pràctiques eficients, primer ens haurem de fixar en la pròpia definició de pràctiques, definint objectius que persegueixen i la seva metodologia.

Objectiu i destinataris

- Proporcionar un complement a la formació acadèmica i contribuir a l'orientació professional dels futurs llicenciats. Les pràctiques van adreçades a alumnes de les assignatures de Sistemes Operatius de la Universitat Autònoma de Barcelona.
- Dotar d'un sistema de suport a la docència per al personal docent del Departament d'Arquitectura de Computadors i Sistemes Operatius (DACSO)

Metodologia

- Les pràctiques consisteixen en la realització de diferents propostes d'exercicis en la que cada alumne haurà de dedicar una hora determinada d'hores. Durant aquest període, els alumnes hauran d'assistir un conjunt d'hores de forma

presencial en els laboratoris del centre i les següents hores restants l'alumne haurà de dedicar de forma no presencial a la resolució del problema descrit.

- En finalitzar les pràctiques, els alumnes hauran de presentar al seu professor de pràctiques una memòria valorativa, exposant el mètode de resolució utilitzat, així com la impressió del codi realitzat. Per tal de facilitar una millor correcció s'entregarà un qüestionari que cada alumne haurà de complimentar, les preguntes del qual seran exposades per el professor.

Durant els cursos anteriors es van realitzar un conjunt de pràctiques, sovint les mateixes, en les que l'alumne podia experimentar nocions bàsiques de les assignatures de Sistemes Operatius.

Aquesta implementació es basa en el desenvolupament d'unes funcionalitats base:

- Gestió de pràctiques (Entorn web d'administració.)
- Desenvolupament de simuladors.
- Disseny i desenvolupament d'un nou conjunt de pràctiques.

Totes aquestes funcionalitats seran explicades en el capítol de desenvolupament (capítol 4 d'aquesta memòria), on es veuran en detall les possibles utilitats i avantatges que cada un d'ells poden aportar al camp de l'ensenyament.

Mitjançant el projecte es pretén crear un model de plataforma que ens pugui ajudar el màxim possible a poder dur a terme, amb la major eficàcia i solvència, els nous reptes que ens plantegen les assignatures tractades i posar en pràctica les noves metodologies d'aprenentatge. Per tal d'assolir-lo, podríem organitzar el primer objectiu del projecte en quatre vessants:

- Estudi previ de les pràctiques impartides fins el moment al centre i visió general del sistema de pràctiques en diferents universitats espanyoles.
- Definició de requeriments de dades i funcionals que es volen
- Disseny i desenvolupament de les noves pràctiques.
- Comprovar la usabilitat del sistema gestor de pràctiques.

En els següents capítols explicarem detalladament tot el desenvolupament del projecte. Al capítol 2 es detallen els principals requeriments que hem plantejat, requeriments de dades i requeriments funcionals, buscant que siguin suficientment amplis per poder cobrir les necessitats de les diferents metodologies docents i les diferents assignatures. El capítol 3 és una explicació i justificació de l'elecció de desenvolupar un nou entorn de gestió de pràctiques. El capítol 4 conté l'explicació del desenvolupament de l'entorn, on s'explica el procés d'anàlisi i implementació del sistema. Els capítols 5 i 6 contenen tota la informació sobre el nou entorn de treball (el minikernel) sent aquests la referència per al desenvolupament de totes les pràctiques proposades. Les pràctiques amb el minikernel estaran contingudes dintre del capítol 7 seguit d'una proposta d'una pràctica clàssica en shell script en el capítol 8. Per finalitzar, al capítol 9 s'expliquen les conclusions obtingudes un cop acabat el projecte.

2. ANÀLISIS DE REQUERIMENTS

A partir de les necessitats de les diferents assignatures de Sistemes Operatius, s'ha fet un estudi dels requeriments, tant per part del disseny de les pràctiques com les funcionalitats del gestor repositori que les administrarà (requeriments gestor-web).

Per una millor comprensió i realització del projecte s'ha dividit el conjunt de requeriments, de dades i funcionals, en dos capítols:

- Requeriments de Disseny i Dades
- Requeriments Funcionals

2.1 Requeriments de Disseny i Dades

Aquí inclourem tots els supòsits per a la elaboració de noves pràctiques:

Es triarà en la seva majoria l'entorn Linux per el desenvolupament de les pràctiques.

L'estructura del desenvolupament anirà clarament lligat al temari de l'assignatura.

Les pràctiques seran graduals i entre si hi haurà una certa dependència per a motivar que els alumnes no hagin de fer grans salts en la comprensió de pràctiques successives.

L'estudi del entorn proporcionat per la Universitat Politècnica de Madrid ens aportarà un nou àmbit d'actuació on aprofundir sobre el disseny dels Sistemes Operatius.

S'elaboraran diferents simuladors corresponents al temari per a la comprensió directa dels continguts vistos en teoria.

El sistema gestor de pràctiques ha estat desenvolupat en php, utilitzant javascript sota un servidor Apache funcional tant en el sistema operatiu Windows com Linux.

2.1.1 Material docent i material per l'alumnat

Les pràctiques impartides contindran varis formats, la versió per l'alumnat en la que com a màxim es proporcionarà un esquelet del programa i l'enunciat del problema a resoldre, i la versió documentada i resolta que podran disposar tots els membres del professorat de l'assignatura.

Estructura material per l'alumnat

- **Nom de la Pràctica**
- **Temes requerits per a la elaboració de la pràctica.**
- **Hores estimades per a la realització de la pràctica.**
- **Data de publicació i finalització de l'avaluació** . Indicarà a l'alumne els terminis d'entrega de la pràctica en la que abans del dia fixat la pràctica hauria d'haver estat ja avaluada.

- **Descripció del problema a resoldre.** Amb exemples de sortides correctes del problema descrit per a facilitar la comprensió del que realment s'està demanant.

Estructura material per al professorat

- **Nom de la Pràctica**
- **Temes requerits per a la elaboració de la pràctica.**
- **Hores estimades per a la realització de la pràctica.**
- **Autor, creador de la pràctica**.
- **Descripció del problema a resoldre.** Amb exemples de sortides correctes del problema descrit per a facilitar la comprensió del que realment s'està demanant.
- **Solució documental de la pràctica.** En format text indicant i fent referència als passos seguits per a la resolució del problema i indicant els punts vitals a tractar.
- **Arxius font que resolen el problema exposat.**
- **Altres arxius.** El professorat podrà disposar de tots aquells arxius que creguin que poden aportar quelcom per a la resolució o per a la explicació del problema.

2.2 Requeriments Funcionals (sistema gestor de pràctiques)

Per a descriure aquests requeriments, hem de tenir en compte els tipus d'usuaris de l'aplicació i les funcionalitats que podrà fer cada tipus d'usuari: Professors, becaris i administrador. Per això els continguts d'aquesta secció seran:

- Funcionalitats dels professors.
- Funcionalitats dels becaris
- Funcionalitats de l'administrador

2.2.1 Funcionalitats dels professors

No diferenciarem entre diferents graus de professorat i la informació que podrà accedir un professor adjunt o un professor responsable de l'assignatura seran les mateixes. Tot seguit explicarem els diversos rols que aquest usuari és capaç d'utilitzar segons els privilegis que l'usuari administrador li hagi ofert.

Els professors responsables estaran assignats a assignatures, i dintre d'elles podran crear i administrar el contingut de temari que s'impartiran, tenint accés al repositori de pràctiques de les assignatures que en siguin responsables.

Dins del menú temari correlacionat directament amb cada assignatura disposaran de les següents funcionalitats:

Temari

- Creació de temes principals
- Crear branques dependents de l'arrel del temari, construint un arbre de desenvolupament de temes relacionats amb com a màxim 3 nivells de profunditat.
- El contingut del temari seran la base de construcció del sistema repositori i constituiran les carpetes on s'emmagatzemaran les pràctiques.

Afegir noves pràctiques

- Els usuaris amb rol professor són capaços d'introduir noves pràctiques en el sistema repositori, que seran visibles per els altres usuaris amb rols becari o professorat que tinguin accés a l'assignatura.

Sistema de pràctiques

- El sistema de pràctiques correspon al repositori del conjunt de pràctiques de la assignatura escollida, on s'emmagatzemaran tot el conjunt de pràctiques accessibles per la resta d'usuaris del sistema.

2.2.2 Funcionalitats del rol becari

El rol Becari està definit com aquell membre docent de l'assignatura (sovint amb una tassa alta de rotació) que no té accés a la nova incorporació de noves pràctiques i que el ús docent que pot fer de l'aplicació és purament de consulta i/o referència per a l'ajuda al suport de l'assignatura en qüestió.

Les funcionalitats del rol Becari seran les mateixes que un professor sense la possibilitat de esborrar arxius, modificar temaris, ni incorporar material.

Temari

- Consulta dels temes principals de l'assignatura.

Sistema de pràctiques

- La funcionalitat que el usuari becari pot fer del repositori es limita a un ús purament consultiu, ja que com em comentat, aquest tipus de rol està específicament lligat als membres que ofereixen suport a l'ensenyament limitant-se a les col·laboracions d'hores lectives dintre dels laboratoris de pràctiques.

2.2.2 Funcionalitats del rol administrador

Per el manteniment d'usuaris, creacions de noves assignatures, assignacions de privilegis usuaris-assignatures es necessita un usuari supervisor, deslligat de les funcionalitats lectives.

L'usuari administrador serà capaç d'administrar els següents punts :

Assignatures

- Donar d'alta noves assignatures al sistema, que podran ser assignades per a ser administrades i consultades per els rols professor/becari.
- Activació o Desactivació de la publicació de l'assignatura, una assignatura podrà deixar de ser consultada però es conservarà al sistema per a una futura utilització.
- Eliminació d'assignatures, d'aquesta manera s'esborraran tots els arxius lligats a tots els temaris lligats a aquesta assignatura eliminant tot el contingut permanentment del sistema.
- Veure usuaris assignats per assignatures.

Usuaris

- Donar d'alta nous usuaris, aquests podran assolir els rols professor/becari/administrador depenent de les necessitats que es requereixin.
- Un usuari podrà formar part del sistema però ser desactivat temporalment, d'aquesta manera l'usuari en qüestió només podrà accedir quan l'usuari administrador ho cregui oportú.
- Eliminació d'usuaris.
- Visualització del sistema de Logs, on podrà contemplar els esdeveniments del sistema, accessos , esborrat d'arxius, etc., amb tota classe d'informació sobre qui ha fet l'acció i des de quina màquina s'ha fet per a poder contemplar qualsevol ús fraudulent del sistema. El Log sencer correspon a un fitxer auto generat pel sistema que conté tots els moviments en format txt.

3. SISTEMA GESTOR DE PRÀCTIQUES – Aplicació Web

3.1 Per què la elaboració d'un sistema gestor de pràctiques?

La idea principal del sistema repositori va sorgir principalment de la idea comuntment utilitzada en el nou pla de Bolonya del portfoli, es pretenia donar d'una manera ordenada i clara els continguts que aquest projecte podia atorgar al departament de la qual pertany l'assignatura de Sistemes Operatius el DACSO.

Donada que la organització de les pràctiques no estava centralitzada cada cop el Sistema Gestor de pràctiques es va contemplar com una gran eina de la que molts professors, becaris i altres membres de qualsevol departament en podria fer un gran ús i donar una eina de la qual no se'n disposava.

El sistema per damunt de tot és un sistema senzill, en alguns cassos purament consultiu, però que satisfà tot un conjunt de necessitats que havien quedat en l'oblit.

Amb el nou sistema de control de pràctiques, aquestes poden ser accessibles des de qualsevol ordinador tan pot ser des de Internet o xarxa interna de la facultat, convertint i centralitzant un sistema obsolet d'emmagatzemament de pràctiques en un repositori ordenat i accessible des de qualsevol facultat adscrita en la que algun membre (actiu i amb permisos) del departament en requereixi el seu accés i ús.

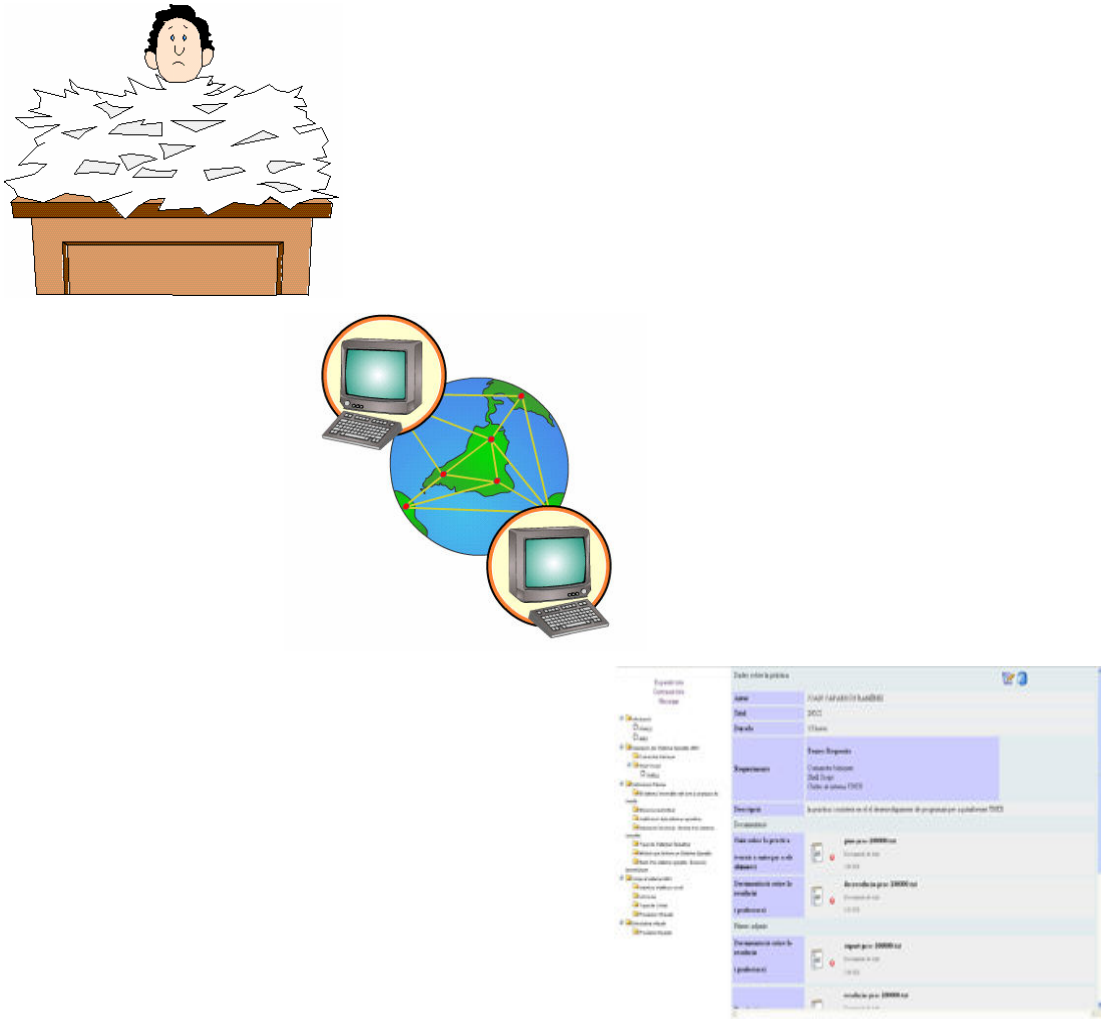


Figura : Representació de les aportacions del nou sistema repositori

4. DESENVOLUPAMENT

4.1 Mòduls i estructura de la Base de Dades

En aquest apartat mostrarem quines han estat les implementacions que hem hagut de realitzar, per aconseguir els requeriments, tant funcionals com de dades, desitjats en el projecte.

Caldrà diferenciar l'elaboració dels mòduls funcionals d'ús acadèmic i l'ús i mòduls del usuari administrador que s'explicaran més endavant.

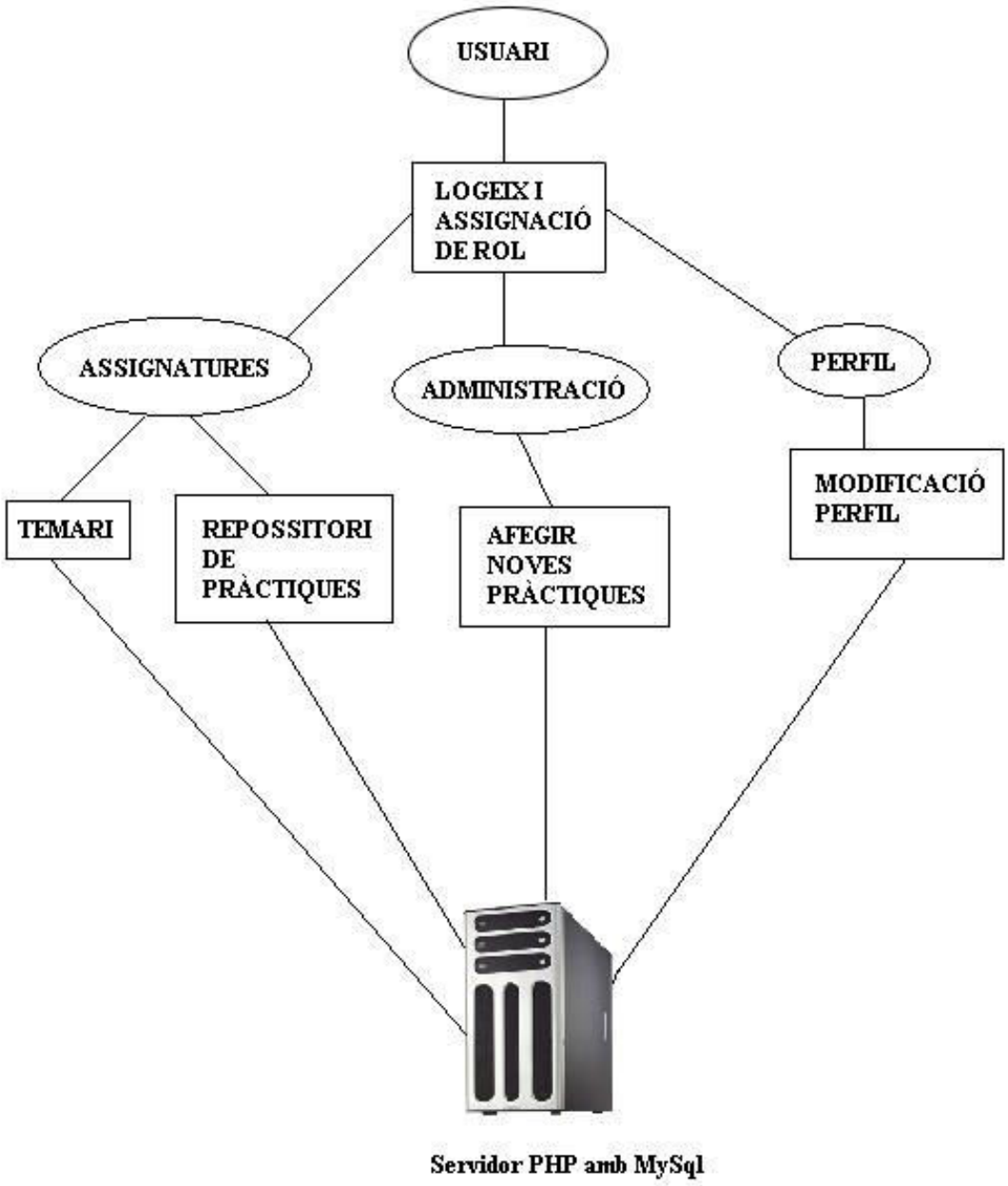


Figura :Diagrama de mòduls i funcionalitats del Sistema Gestor de Pràctiques

Un cop presentats els mòduls nuclis del sistema presentarem l'estructura de la base de dades :

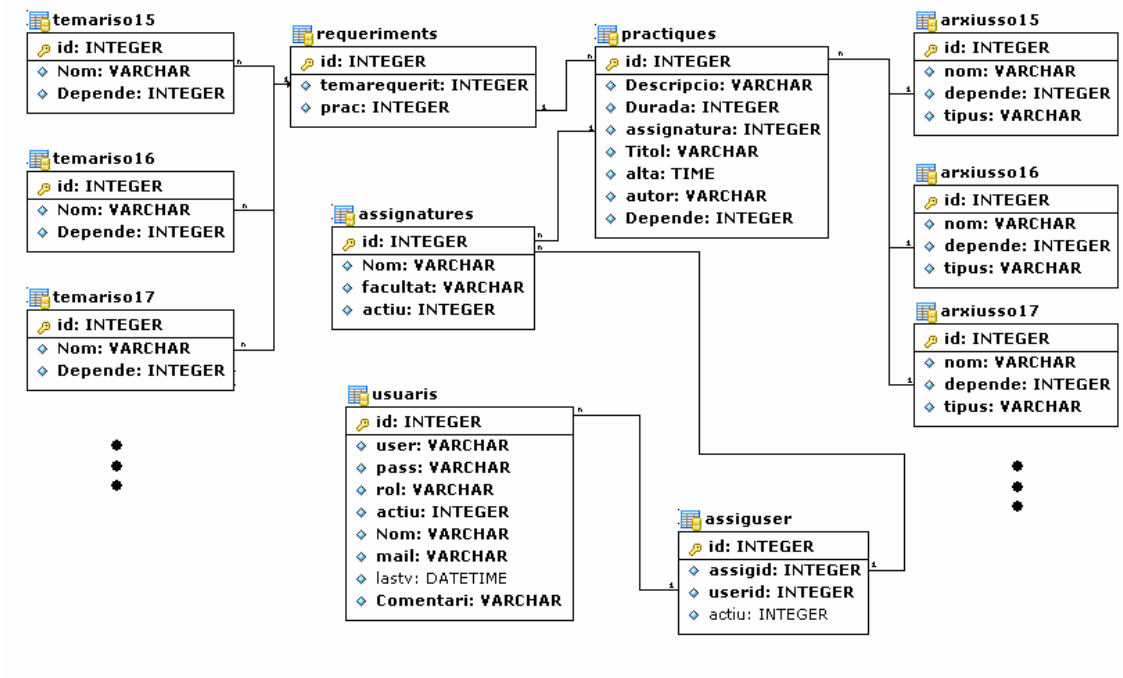
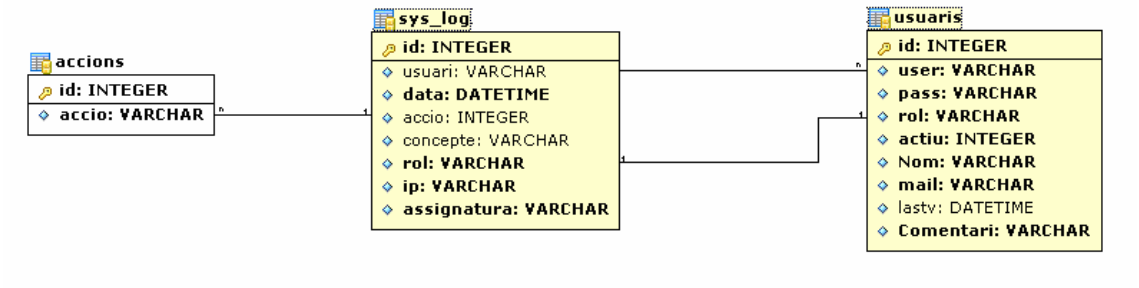


Figura :Esquema de la interrelació entre taules MySql



Construcció MySql del sistema de log

4.2 Pàgina d'inici

Com és normal en la filosofia de la construcció de pàgines web aquesta pàgina correspon al fitxer ***index.php*** la seva construcció és senzilla i es limita a rebre el nom d'usuari i el password corresponent per a logar els usuaris dins el sistema.

Figura :Pantalla d'inici referent al logeix d'usuaris

Un cop l'usuari ha introduït el seu nom i contrasenya (que li hauran estat comunicats per el usuari administrador) el fitxer ***validar_usuario.php*** verificarà :

- Si l'usuari ha introduït una contrasenya valida, en cas contrari tornarà a carregar la pàgina d'inici ***index.php.***

- Si la contrasenya correspon amb la atorgada al usuari, el sistema generarà una *cookie* xifrada amb informació sobre el rol (becari / professor / administrador), registrarà l'entrada en el fitxer de log i redireccionarà cap a la pàgina d'inici de la aplicació.

- Al accedir a la pàgina d'inici aquesta genera una *cookie* no xifrada de nom *"layera"* que capturà mitjançant javascript la resolució de la pantalla en la que s'està treballant per tal de readaptar les pàgines. Inicialment l'aplicació fou realitzada a una resolució de 1280 i readaptada a 1024 mitjançant la generació d'aquesta *cookie.*

- Redireccionament depenent del rol :

  - Si l'usuari posseeix el rol *becari* o professor la pàgina de validació redireccionarà a l'usuari cap al fitxer ***inici.php*****,** que contindrà l'esquelet de construcció de les pàgines i que només afegirà modificacions al diferenciar els rols.

  - Si al validar-se aquest usuari conté privilegis d'administrador la plana visualitzada correspondrà amb el fitxer ***admin.php***, amb totes les funcions especifiques per a administrar assignatures, usuaris i moviments realitzats dintre del sistema.

4.3 Estructura de les pàgines

Definirem 3 zones contingudes en qualsevol pàgina del sistema :

- **Zona de Menús** :

  Els menús es situen a la zona superior de la web, en el apartat ***Els menús*** d'aquesta memòria s'explicaran en tot detall els diferents nivells que aquests poden disposar

- **Àrea de deslogeix** :

  Tot usuari logejat manté la cookie de sessió mentre aquest no es deslogeji, per tant és recomanable que aquest usuari premi el link per deslogejar-se per eliminar aquesta *cookie,* evitant així usos fraudulent del sistema.

  De la mateixa manera en aquesta àrea es visualitzarà el nom complet del usuari i el rol que el sistema li ha assignat.

- **Àrea d'informació** :

  Àrea central on es mostra el contingut central del apartat seleccionat en la Zona de Menús.

Figura :Estructura bàsica d'informació i Menús en la web

4.4 Els menús

Els menús de l'aplicació i l'accés a les diferents pàgines són les principals diferències entre els diferents rols, com em comentat anteriorment ja que els usuaris tenen diferents permisos d'accés a les planes web.

A continuació mostrarem els diferents menús i entrarem en detalls sobre l'accés que aquests poden oferir.

4.4.1 Usuari Becari

El menú del usuari becari conté els següents apartats :

- **Inici**: El Sistema Gestor de pràctiques conté una pàgina d'inici de presentació del projecte, totalment modificable per a deixar un espai a la presentació del sistema als seus nous usuaris.

  Des de ell l'usuari becari podrà:

  - Tornar ràpidament a la pàgina principal

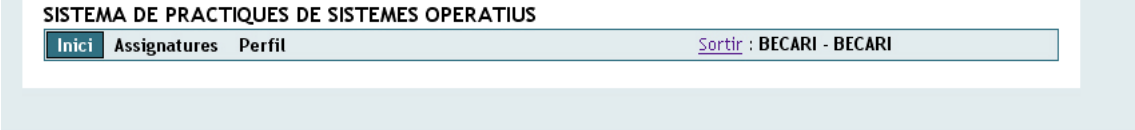


Figura :Menú Usuari Becari - Inici

- **Assignatures** : Els usuaris disposaran d'un menú amb totes les assignatures que tenen assignades, de les quals podrà consultar-ne el temari i el seu repositori de pràctiques. El numero i la disposició d'aquestes assignatures vindrà determinada dels privilegis que l'usuari administrador li hagi atorgat.

  Des de ell l'usuari becari podrà:

  - Consultar assignatures adscrites
  - Consultar temaris
  - Entrar al Sistema repositori

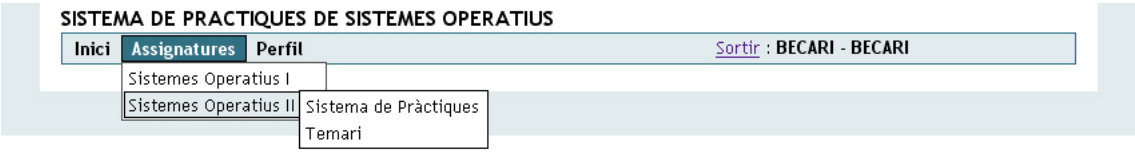


Figura : Menú Usuari Becari - Assignatures

- **Perfil :** Tot usuari pot modificar les seves dades personals, Nom i Cognoms i la contrasenya per accedir al sistema clicant sobre del apartat del menú Perfil. L'usuari d'accés li serà subministrat per l'usuari administrador junt amb la primera clau d'accés que podrà ser posteriorment modificada.

  Des de ell l'usuari becari podrà:

  - Modificar Nom i Cognoms
  - Modificar la contrasenya d'accés al sistema

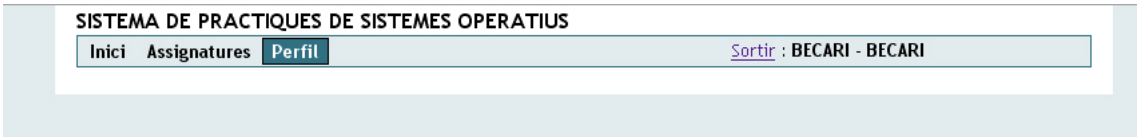


Figura : Menú Usuari Becari - Perfil

4.4.2 Usuari Professor

El rol desenvolupat per els professors requereixen d'una participació activa en les tasques d'incorporacions de nou material al repositori, ja que els usuaris amb rol *professor* són els únics que poden afegir material docent al repositori.

Tots els professors disposaran en el menú de les opcions comentades en l'apartat *Usuari Becari* amb el tret principal de modificar i crear en l'apartat temari i amb la incorporació d'una nova possible selecció (que fa referència directe sobre el fet de l'administració de fitxer) :

- **Administració :** Originalment en el Sistema Gestor de pràctiques els professors només podran escollir la possibilitat d*'afegir noves pràctiques*, però el sistema està preparat per a que es puguin incorporar tots els menús d'administració que els usuaris professors puguin requerir-ne.

  Des de ell l'usuari professor podrà:

  - Afegir noves pràctiques

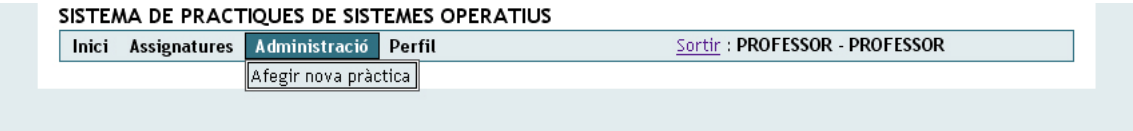


Figura : Menú Usuari Professor - Administració

4.4.3 Usuari Administrador

Els menús referents al usuari administració tot i que en principi puguin fer pensar una mateixa utilització respecte els usuaris becari i professor, aquests són totalment diferents i no són desplegables ja que les àrees a administrar són molt concretes i es limiten a administració d'usuaris i administració d'assignatures.

Tenint accés a aquests diferents apartats d'administració :

- **Assignatures :** Des de el menú *Assignatures* els usuaris amb rol administració podran accedir a tot el conjunt d'assignatures insertades en el Sistema Gestor de pràctiques i visionar-ne quants usuaris estan adscrits.

  Tant mateix podran desactivar assignatures com accedir a les opcions de creació i esborrat.

  Des de ell l'usuari professor podrà:

  - Crear noves assignatures
  - Esborrar assignatures existents.
  - Activar o desactivar estat de les assignatures
  - Visualitzar numero i usuaris amb permisos sobre l'assignatura

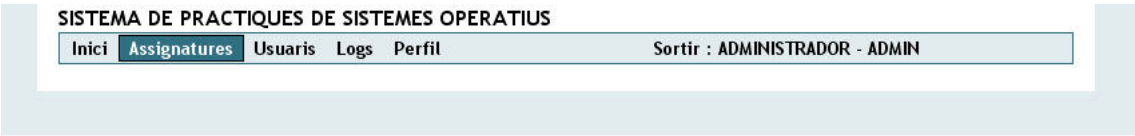


Figura : Menú Usuari Administrador - Assignatures

- **Usuaris:** El control d'usuaris és una peça clau per a la assignació de personal assignat a cada assignatura.

  Des de ell l'usuari administrador podrà:

  - Crear nous usuaris
  - Esborrar usuaris
  - Desactivar temporalment l'accés a l'aplicació a determinats usuaris
  - Atorgar privilegis i assignar o modificar rols

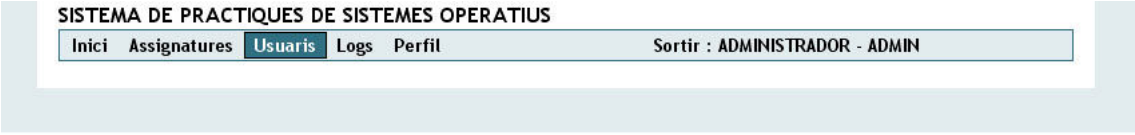


Figura : Menú Usuari Administrador - Usuaris

- **Logs:** Totes les accions dels usuaris queden enregistrades al sistema, d'aquesta manera podem tenir un control més precís sobre qui ha fet quina acció, quan i des de on.

  Des de ell l'usuari administrador podrà:

  - Visualització dels moviments dintre del Sistema Gestor de Pràctiques
  - Accedir al fitxer ***log.txt*** que conté tots el moviments en format arxiu de text.

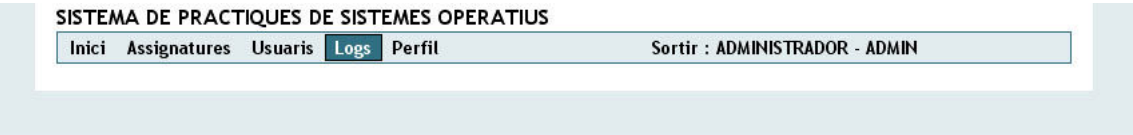


Figura : Menú Usuari Administrador - Usuaris

4.5 Pàgina Principal

Tot sistema ha de tenir una pàgina inicial després del logeix, en aquest cas el Sistema Gestor de Pràctiques disposa d'una plana de presentació amb les dades sobre l'assignació, tutor i dades referents al projecte.

Per a un futur ús aquesta pàgina és completament modificable sent el fitxer ***contingut.php*** l'únic fitxer que s'hauria de modificar, ja que aquest conté el cos central de la estructura de la pàgina principal.

A continuació mostrarem la pàgina principal original del projecte:

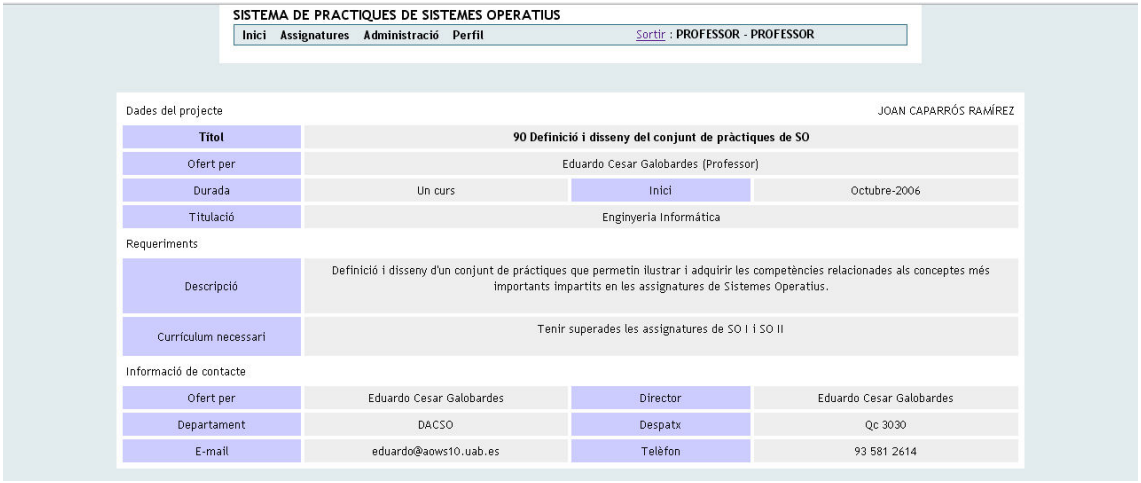


Figura : Pàgina principal

4.6 Assignatures

En l'apartat *Zona Menús – Assignatures* em fet una petita referència sobre les capacitats que els usuaris poden desenvolupar dins de l'apartat Assignatures, sent aquesta una àrea d'accés a les diferents assignatures que tenen assignades tan per consulta de temaris o per accedir al seu repositori de fitxers.

Dos rols són capaços d'accedir a aquests recurs però amb resultats molt diferents. En cada un dels cassos explicarem quines són les utilitats o funcionalitats que en ells poden assolir, per tant es subdividiran els pròxims apartats en utilitats del rol *becari* i del rol *professor*.

4.6.1 Temari

Usuari Becari : Els usuaris amb rol becari poden accedir a tots els temaris de les assignatures que estan adscrits. L'ús que en poden fer és purament consultiu.

La seva implementació es troba subdividida en tres fitxers, un encarregat de la construcció de la pàgina, amb l'esquelet de funcions ***temaris.php,*** zona de menús ***menu.php*** i el contingut del temari ***temari-becari.php.***

A continuació mostrarem la pantalla referent al temari de la assignatura de Sistemes Operatius I consultada per un usuari amb rol becari :

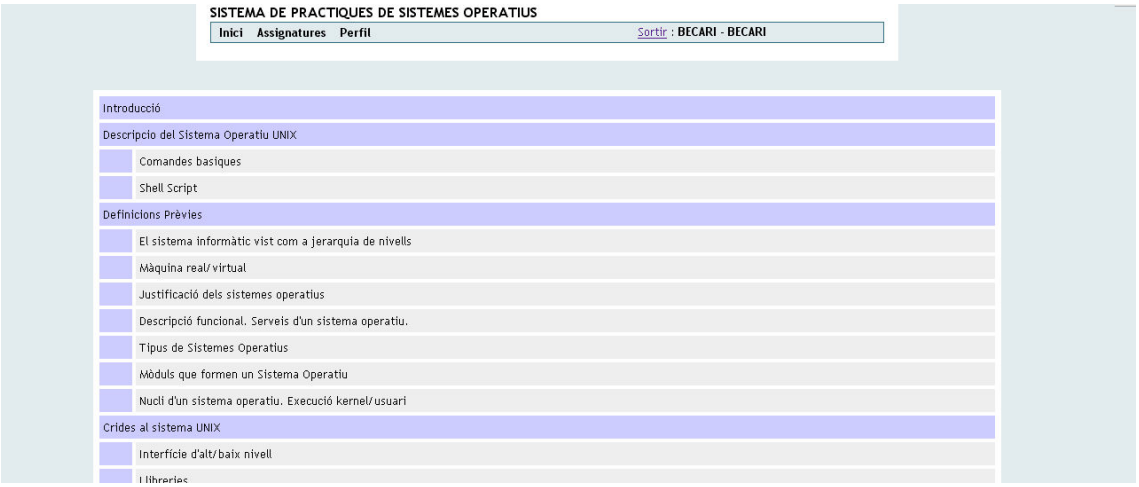


Figura : Pàgina consulta temari.

Usuari Professor : Tot professor és un membre actiu en la incorporació o modificació dels temaris impartits en les seves assignatures, per tant només ells podran accedir a la modificació de temaris. En el següent apartat on es farà referència al Sistema de pràctiques (sistema repositori) veurem la gran influència que l'apartat *temari* efectua sobre aquest repositori.

La construcció d'aquesta pàgina web es bassa en la mateixa estructura que la plana de consulta de temaris relacionats amb el rol becari, exceptuant que aquest s'ha optat per desenvolupar el seu contingut en un fitxer separat de nom ***temari-prof.php***.

Abans de qualsevol explicació referent al comportament de la plana mostrarem una imatge del seu contingut :

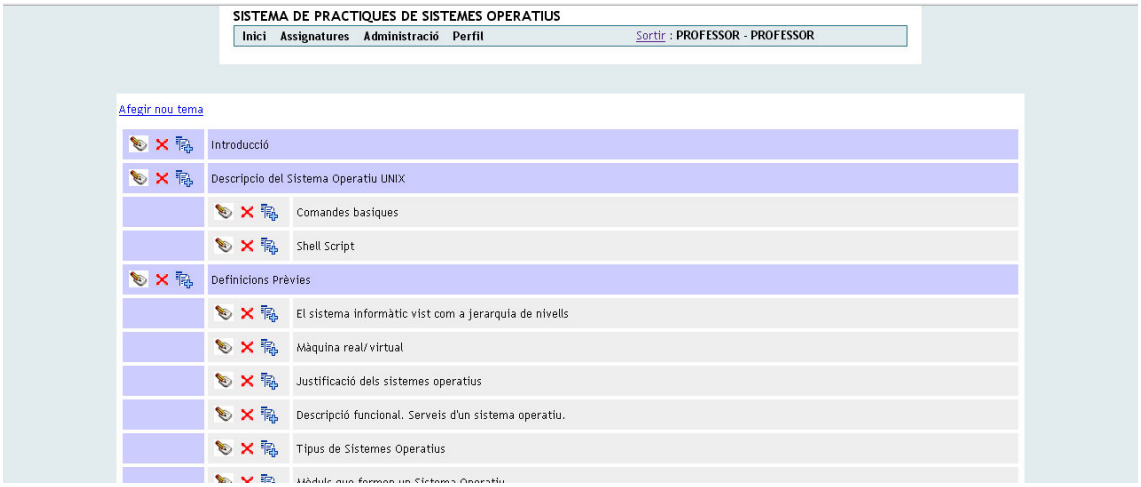


Figura : Pàgina temari usuari professor.

En l'anterior imatge és pot apreciar de forma visual els diferents nivells que poden tenir cada apartat, podent crear temes principals i temes relacionats entre ells amb tres nivells de profunditat com a màxim.

És poden efectuar diferents accions sobre els temaris:

- Afegir un nou tema principal, del qual podran dependre subtemes, clicant sobre el link amb el mateix nom.
-  / Afegir subtema : El següent icona fa referència a la incorporació de subtemes relacionats amb el tema en el que em polsat la imatge. Un cop em clicat obrirà un nou subtema del que podrem assignar-li un nom i enviar-lo o cancel·lar l'opció i tornar al temari original.

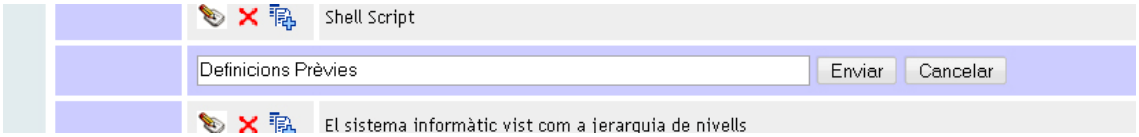


Figura : Pàgina temari incorporació subtemes.

-  / Modificar : El sistema contempla els errors que es puguin produir a la hora d'introduir dades al sistema. Al clicar, el nom del tema o subtema es tornarà editable i permetrà una modificació del nom que figurava a la base de dades.
-  / Eliminar : Tot tema o subtema es eliminable, però cal remarcar que de la eliminació d'un subtema és produeixen els següents afectes sobre el sistema repositori de pràctiques:
    - **Modificació del temari :** el tema o subtema en qüestió deixa d'aparèixer en el temari de l'assignatura .Un esborrat accidental podria ser molt perjudicial, qualsevol intent de esborrat serà confirmat mitjançant la funció *confirm* que ens ofereix javascript, quedant un quadre de confirmació com el següent :

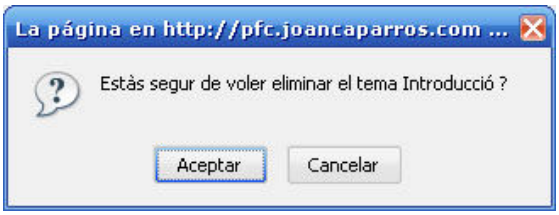


Figura : Quadre confirmació d'eliminació de tema/subtema "Introducció".

El mecanisme d'eliminació farà les següents comprovacions:

1 - El tema no ha de contenir cap pràctica lligada

2 - No ha de tenir subtemes lligats, en cas de voler esborrar una branca sencera s'hauran d'eliminar totes les fulles per a poder esborrar el tema principal.

En el cas que el sistema detecti algun incompliment d'aquestes dues normes ens avisarà amb un missatge.

4.6.2 Sistema de Pràctiques

El repositori de Pràctiques és el nucli del Sistema Gestor de pràctiques, sent aquest l'encarregat de mantenir emmagatzemades de forma correcte i ordenada tota la documentació de cada pràctica.

4.6.2.1 Estructura de la pàgina del Sistema de pràctiques

Es diferencien dues zones diferenciades per ubicar-se en diferents iframes de la pàgina, en el primer iframe trobarem la zona de selecció de pràctiques on apareixeran en forma de carpetes contenidores els temes introduïts en el temari de la assignatura, en el segon iframe contindrà tota la informació rellevant i mostrarà tots els fitxers que el repositori disposa d'una pràctica en concret.

La següent imatge il·lustrarà de forma gràfica la posició d'aquests dos apartats:

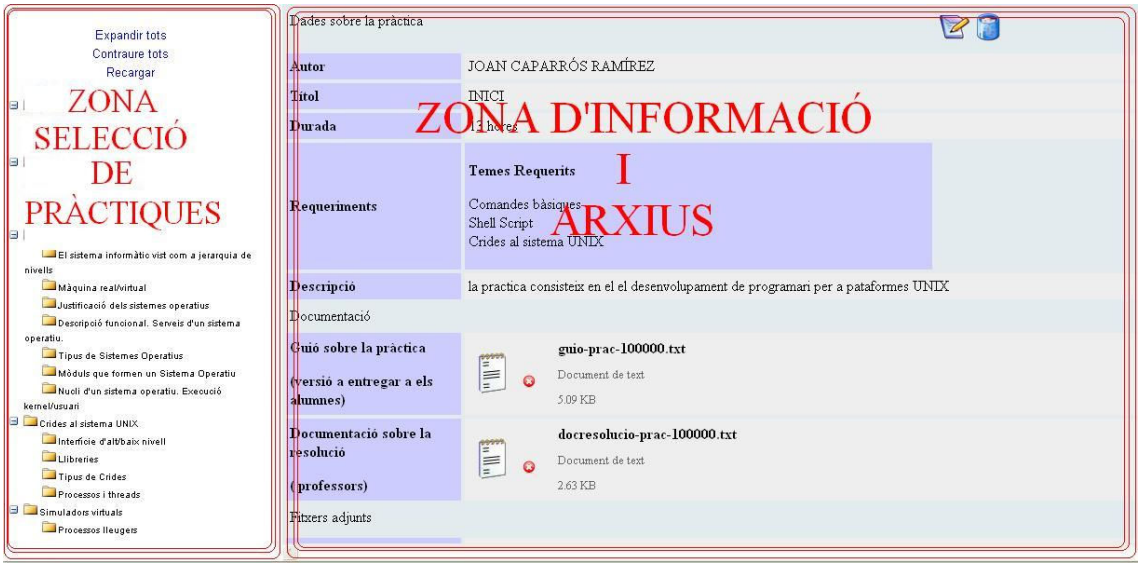


Figura : Estructura iframes Sistema de pràctiques.

Cada una de les zones tenen diferents comportaments i permeten diferents accions, degut això en aquest apartat separarem les funcionalitats de les dues tot i que, com ja veurem interaccionen al efectuar accions.

Zona selecció de pràctiques

Per descriure la següent zona tornarem a fer referència al apartat temaris de la zona de menús de professors i becaris (apartat 4.6.1 d'aquesta memòria), cada tema introduït en el apartat temari serà incorporat a la zona de selecció de pràctiques com un contenidor del qual poden dependre les diferents pràctiques.

Per exemple, si a la assignatura de sistemes Operatius I s'incorpora el tema principal *"Descripció del Sistema Operatiu UNIX"* amb subtemes dependents *"Comandes bàsiques"* i *"Shell Script"* aquest serà visionat com un conjunt de carpetes dependents on es podran incorporar assignatures lligades a aquest tema o subtema "carpeta"

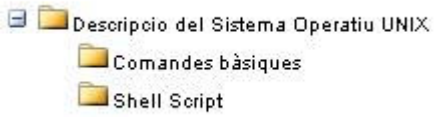


Figura : Exemple construcció de carpetes i mostratge.

Un cop s'ha denotat el mode de visionat de les carpetes contenidores de pràctiques proposarem la introducció d'una pràctica relacionada en una d'aquestes carpetes (temes o subtemes).

Propossarem introduir una pràctica dependent del temari de *"Shell Script"* amb títol *"pràctica 1",* aquesta última apareixerà com una fulla d'aquesta carpeta que serà seleccionable per a visionar-ne el seu contingut (dades relacionades amb informació i arxius). Per tant la construcció resultant d'haver afegit una pràctica a un tema o subtema romandria de la següent manera :

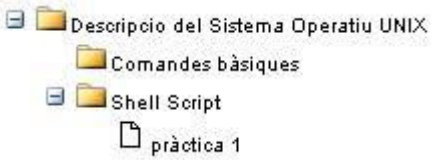


Figura : Exemple afegit de pràctica a carpeta

El mode d'introducció de pràctiques serà comentat al apartat 4.7.1 d'aquesta memòria.

Per finalitzar, un cop les carpetes han estat construïdes i contenen pràctiques podrem disposar de tres funcions que ens permetran treballar amb més facilitat:

- **Expandir tots :** Aquesta opció ens servirà per desplegar totes les carpetes i visionar-ne el contingut sencer, de tots els temes o subtemes de l'assignatura.
- **Contraure tots :** De la mateixa manera la funció Contraure tots ens permetrà "recollir" totes les carpetes per poder tenir d'una forma més ordenada el conjunt de temaris.

- **Recarregar :** Com varis usuaris poden fer servir l'aplicació a la vegada s'ha incorporat aquesta opció per a visionar noves pràctiques introduïdes després del accés al sistema repositori.

Zona d'informació i arxius

La següent plana està formada per dues parts, al clicar sobre qualsevol pràctica en aquesta es podrà visionar una primera zona d'informació seguit del mostreig dels arxius que es requereixen per a la pràctica, ja sigui a nivell de docència com a nivell intern de professorat.

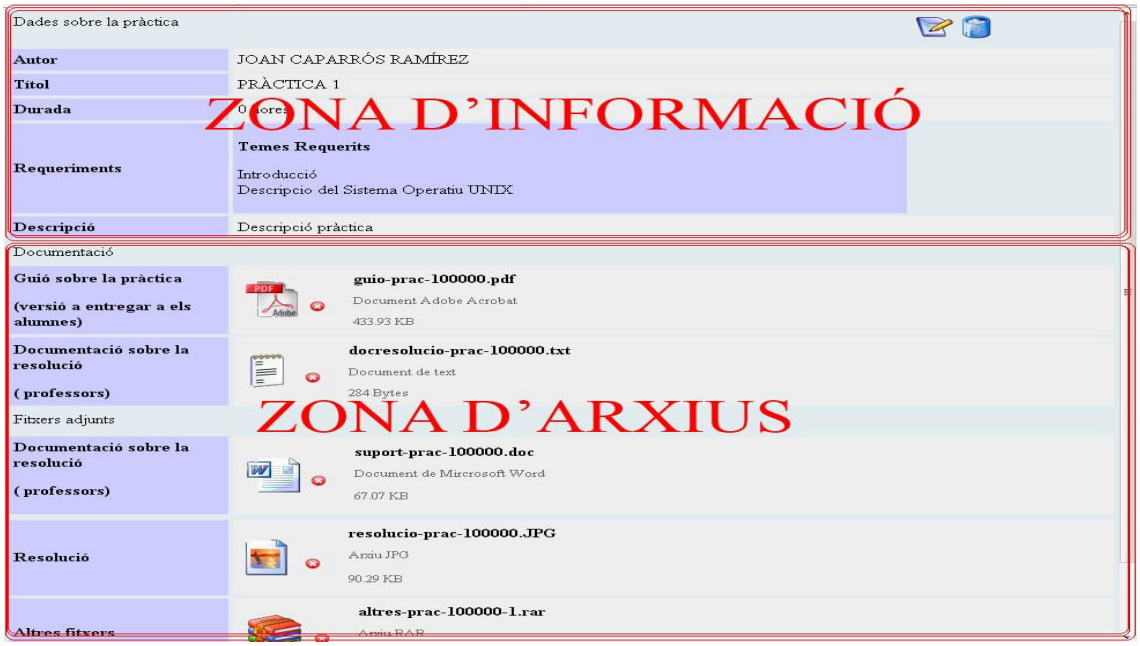


Figura : Estructura Zona informació i arxius.

Per a facilitar la comprensió separarem l'explicació de cada una d'aquestes dues parts que conté la plana :

Zona d'informació : Aquesta Zona permetrà una consulta ràpida sobre :

- Autor : Realitzador de la pràctica
- Títol
- Durada : Estimació d'hores de dedicació.
- Temes Requerits : Temes o subtemes que calen comprendre per a la realització de la pràctica.
- Descripció : Breu comentari referent a la realització o contingut de la pràctica.

Tenint aquesta petita descripció, només caldrà veure si alguna pràctica en concret reuneix els requisits per a ser proposada la seva elaboració per els alumnes, aconseguint una major lleugeresa a la hora de escollir les pràctiques que seran realitzades.

La Zona d'informació ens oferirà dues accions referents a modificació i eliminat de la pràctica. A continuació mostrarem els icones que se'n disposa i l'acció que efectuen sobre les pràctiques.

- / Modificar : Al clicar sobre la opció de modificació el contingut de la plana serà el mateix del de la zona d'informació totalment editable. Permetent una modificació de qualsevol camp que apareixia en el apartat anterior, oferint la possibilitat de recol·locar la pràctica en qüestió en un altre tema o subtema de l'assignatura.
- / Eliminar pràctica : La eliminació de qualsevol pràctica comporta un esborrat permanent de tots els arxius que aquesta en fa referència, per això es un tret important que caldrà reconfirmar, ja que un esborrat fortuït suposaria la eliminació per complet de tot el material docent emmagatzemat fins el moment.

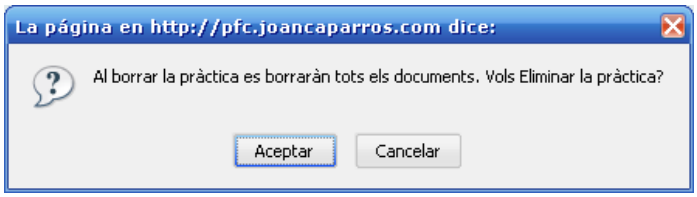


Figura : Quadre confirmació d'eliminació de pràctica.

Zona d'arxius : La Zona d'arxius ens mostrarà tots els fitxers que la pràctica requereix per a la seva elaboració o per a un ús purament didàctic.

En el apartat 4.7.1 s'explicaran en tot detall quina classe de fitxers contempla el repositori.

Per facilitar el visionat dels arxius el sistema captura la seva extensió, tamany i nom mostrant el conjunt de propietats bàsiques del fitxer de la mateixa manera que ho faria el sistema operatiu windows.

Figura : Propietats bàsiques mostrades pel repositori..

El costat de l'arxiu podem observar un icona, aquest ens permetrà efectuar l'única acció que es pot efectuar quan un arxiu existeix:

- / Eliminar arxiu : Tots els arxius pujats poden ser esborrats per qualsevol usuari professor, si aquest creu oportuna la seva eliminació del sistema. Com en altres casos una eliminació de fitxers ha de tenir una confirmació que vindrà expressada com a resultat de la següent finestra :

Figura : Quadre confirmació d'eliminació de fitxer.

Els icones corresponents les diferents extensions seran contemplades de diferents maneres, el sistema originalment aporta icones especifiques per a arxius amb extensió jpg, gif, png, bmp, txt, c, zip i rar sent aquests mostrats de la següent manera :

Icone	Tipus	Extensions referents
	Imatge	png , jpg, bmp, gif
	Text	txt, c
	Document Word	.doc
	Fitxer comprimit	zip, rar

Usuari Becari : L'usuari becari pot accedir al sistema repositori de pràctiques de la mateixa manera com ho faria qualsevol usuari amb rol professor, només caldrà remarcar petits canvis que afectaran al comportament de la plana , l'usuari no podrà modificar, esborrar o aportar documentació limitant el seu ús a la consulta de material.

L'aspecte de la web és molt similar però en aquest cas desapareixen les icones que en el usuari professor feien referència a edicions del material docent.

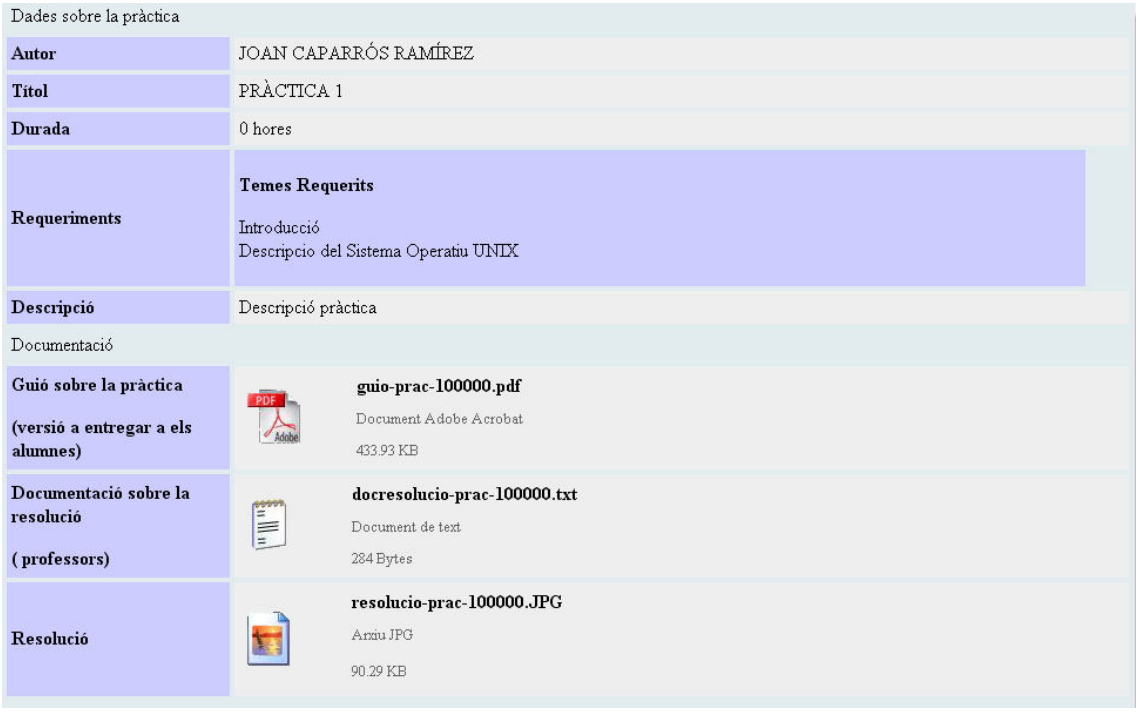


Figura : Estructura Zona informació i arxius.

4.7 Administració

L'aplicació consta d'àrea d'administració lligada al àmbit del professorat responsables de les assignatures. En aquesta primera versió de l'aplicació es dotarà de permisos al usuari amb rol *professor* de afegir noves practiques al repositori, sent només aquest l'únic tipus de rol que s'encarregarà de aportar material docent a la aplicació.

4.7.1 Afegir noves pràctiques

En referència a la capacitat d'incorporació de noves pràctiques al sistema, denotarem els trets bàsics que s'hauran d'especificar per tal d'introduir qualsevol material.

Tota pràctica haurà de complir els següents requisits :

- S'especificaran tant l'autor, títol, durada en hores, una petita descripció, com a quina assignatura vol ser incorporada en el repositori, aquestes dades serviran per emplenar la *zona d'informació* del sistema gestor i per determinar la seva ubicació en el arbre referent a la *zona de selecció de pràctiques* .

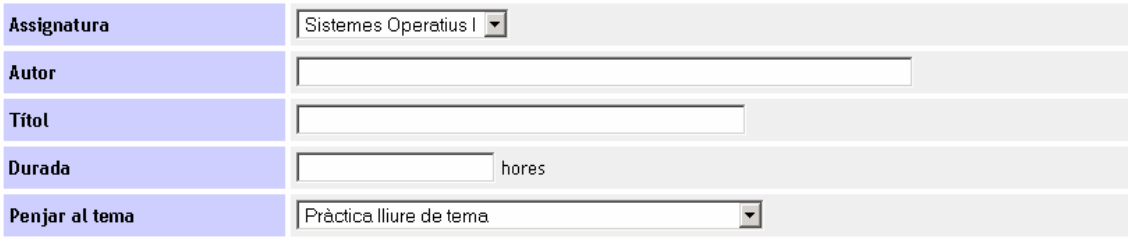


Figura : Afegir noves pràctiques – Àrea d'informació.

- Per a cada pràctica s'inclouran un llistat de temes que l'alumnat ha de conèixer per a poder desenvolupar-la, d'aquesta manera només caldrà mirar quins temes són requerits per a assignar una pràctica o una altre segons l'alçada del curs en la que ens trobem.

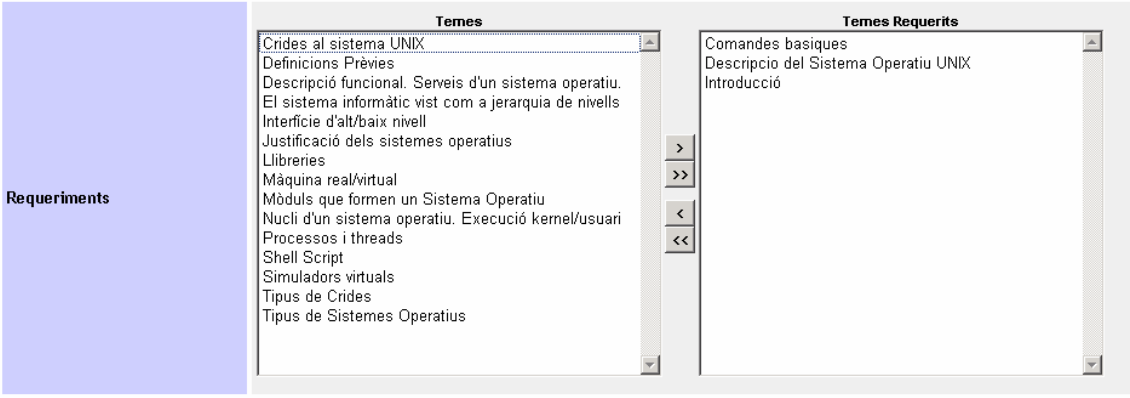


Figura : Afegir noves pràctiques – Àrea d'informació – Requeriments

El seu funcionament és senzill i només caldrà desplaçar la selecció de temes que es creguin convenients recol·locant-los en el *multiplebox* que fa referència a els temes requerits.

- Documentació : Existeixen dos classes de documentació lligades a cada material, l'un farà referència al material d'ús intern del departament i que ajudarà a tot el col·lectiu destinat a impartir les classes de pràctiques en la seva realització, i l'altre farà referència al material que serà entregat directament als alumnes per a la seva realització.
    - **Documentació per al professorat** : Es podran pujar documents específics per a ús exclusiu per al professorat, referents a la resolució de la pràctica.
    - **Documentació per als alumnes** : Els documents amb la informació necessària per a que l'alumnat realitzi la pràctica també seran dipositats en el repositori.

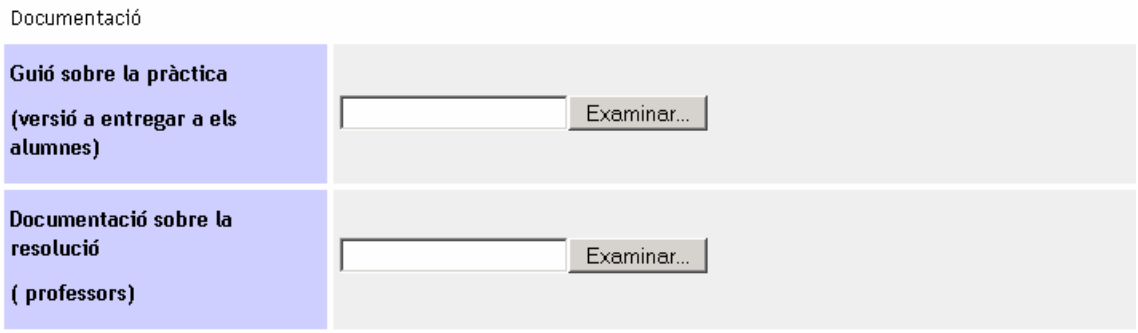


Figura : Afegir noves pràctiques – Àrea d'arxius – Documentació.

- Fitxers adjunts : Els fitxers d'adjunts representen tota aquella documentació referent a codi que s'aporti per a ajudar a la realització o per facilitar la correcció al personal docent.
    - **Fitxers de suport a la pràctica** : El material de suport es específic per a l'alumnat i en el cas que s'aportin, aquest serà un esquelet on hauran de desenvolupar parts especifiques per a desenvolupar problemes concrets.
    - **Resolució :** S'inclourà la possibilitat d'incorporació d'un fitxer (a poder ser comprimit) que contindrà els fitxers que ofereixen una possible resolució de la pràctica a elaborar per a l'alumnat, d'aquesta manera professors que no han estat implicats en el desenvolupament del material tenen una referència sobre el seu desenvolupament.
    - **Altres fitxers** : A part dels fitxers de suport i de resolució el professorat pot creure convenient tenir documentat o emmagatzemat altres informacions referents a la pràctica, aquestes poden ser desades en aquest camp.

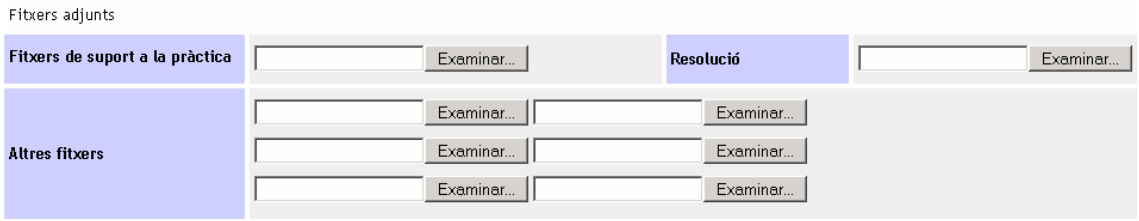


Figura : Afegir noves pràctiques – Àrea d'arxius – Fitxers d'ajut.

4.8 Perfil

Com podrem veure en la utilització del *Sistema Gestor de pràctiques* tots els usuaris disposen de la capacitat de modificar les seves dades, el sistema no contempla dades personals i només farà referència a el nom, cognoms i la compte de correu del usuari.

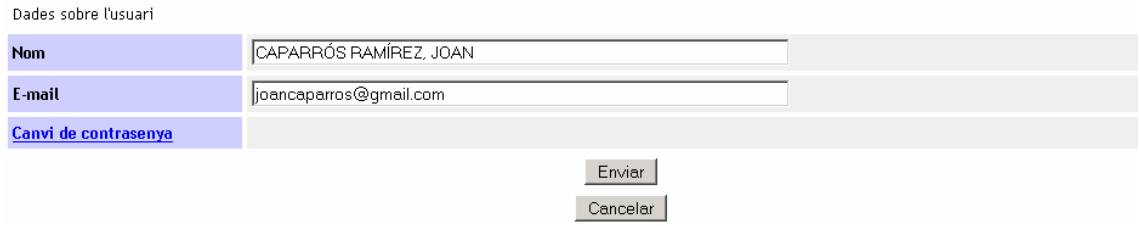


Figura : Pàgina modificació del perfil.

Tant mateix s'ofereix canviar la contrasenya d'accés al sistema, ja que la primera clau serà proporcionada per el usuari administrador i per conservar la seguretat del sistema aquesta hauria de ser canviada.

Figura : Pàgina Perfil canvi de contrasenyes.

4.9 Eines per a la administració del Sistema Gestor

Les eines d'administració pertanyen exclusivament als usuaris amb rol *administrador*, permetent el control de les assignatures actives en el sistema, assignacions de permisos d'accés als diferents usuaris i visionat dels diferents moviments que s'han realitzat per a poder detectar qualsevol ús fraudulent.

4.9.1 Administració d'assignatures

La pàgina **assignatures.php** (Assignatures en el menú) dona la capacitat de :

- Crear noves assignatures : Aquestes vindran representades per el seu nom i per la facultat on s'imparteix.

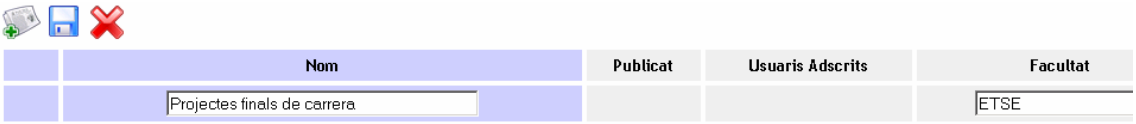


Figura : Creació de noves assignatures.

- Visualitzar el seu de publicació: Cada assignatura pot assolir dos estats [Publicada / No publicada], quan una assignatura està publicada tots els membres que tenen privilegis d'accés poden i per tant visionar-ne el seu contingut, tan mateix si aquesta roman No publicada l'accés estarà restringit temporalment i no apareixerà en els menús dels usuaris *becari/professor* que hi vulguin accedir.
- Eliminar Assignatures : Per un esborrat permanent del contingut introduït en una assignatura només caldrà eliminar-la. Amb aquesta opció eliminarem del sistema tots els arxius pertanyents a la assignatura en qüestió i totes les taules que en feien referència. Com a tots els cassos en el que el esborrat accidental podria suposar grans pèrdues, l'eliminació d'assignatures haurà de ser confirmada.

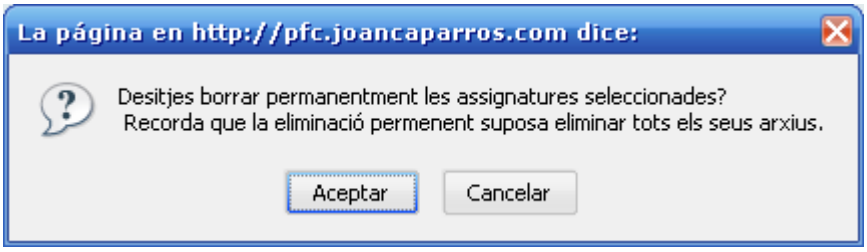


Figura : Confirmació eliminació permanent d'assignatures.

L'aspecte general de l'administració d'assignatures romandria de la següent manera, en una primera columna visualitzaríem el nom de les assignatures, seguit del seu estat de publicació, nombre d'usuaris adscrits (amb privilegis de consulta o incorporació de material) i la seva facultat.

	Nom	Publicat	Usuaris Adscrits	Facultat
☐	Sistemes Operatius	✗	2	Sabadell
☐	Sistemes Operatius I	✓	7	ETSE
☐	Sistemes Operatius II	✓	3	ETSE

Figura : Eina administració d'assignatures.

4.9.2 Administració d'usuaris

El Sistema Gestor de pràctiques conté una eina d'administració d'usuaris, sent aquest l'encarregat d'assignar els rols i privilegis sobre cada un dels membres responsables de les assignatures. A continuació explicarem el conjunt d'accions que es poden realitzar sobre cada usuari :

	Nom	Usuari	Actiu	Rol	E-Mail	Última visita
☐	admin	admin	✓	ADMIN	admin	2007-05-31 09:45:37
☐	ADMINISTRADOR	administrador	✗	ADMIN		2007-05-24 19:49:41
☐	BECARI	becari	✓	BECARI	becari@uab.es	2007-05-24 20:17:40
☐	BLANCO LOPEZ, ALEJANDRO	ablanco	✓	PROFESSOR	jandroblanco@gmail.com	2007-05-02 15:17:59
☐	BLANCO PURO, ALEX	ablanco2	✓	BECARI	fdaf@fcds.es	
☐	BLANCO Y NEGRO, ANTONIO	ablanco3	✓	PROFESSOR	fdaf@fcds.es	
☐	CAPARRÓS RAMIREZ, EDUARD	ecaparros	✗	BECARI	fdaf@fcds.es	
☐	CAPARRÓS RAMÍREZ, JOAN	johan	✓	ADMIN	joancaparros@gmail.com	2007-05-30 15:13:06
☐	CESAR GALOBARDES, EDUARDO	ecesar	✓	PROFESSOR	eduardo@aows10.uab.es7	2007-05-30 15:12:52
☐	PROFESSOR	professor	✓	PROFESSOR		2007-05-26 09:56:27

Figura : Eina administració usuaris.

- Visualitzar el conjunt d'usuaris: La primera columna ens permetrà visualitzar els usuaris donats d'alta al sistema.
- Visualització del usuari d'accés : Mitjançant aquest nom d'usuari cada membre del personal amb la seva contrasenya serà capaç d'accedir al sistema.

- Visualitzar el seu estat d'activació: Cada usuari pot assolir dos estats [Actiu / No actiu], el perquè de l'existència de la possibilitat d'activació d'usuaris recau en el fet que la rotació de personal és bastant elevada i a vegades interessarà conservar usuaris en el sistema però sense haver-los d'eliminar completament, d'aquesta manera un usuari No actiu no podrà accedir al sistema però conservarà tots els privilegis que se li van atorgar en el moment de la seva creació, un usuari actiu podrà accedir normalment al sistema.
- Consultes de Rol, email i última visita : La visualització rapida de rols assignats es podrà fer a través d'aquesta eina juntament amb el email proporcionat i una petita referència a la última vegada que van accedir al sistema.
- Edició d'usuaris i permisos : L'edició d'usuaris consistirà en un formulari on el usuari administrador podrà canviar el nom, el rol (segons les necessitats que requereixi), el email de contacte i una petita descripció per si s'escau. En la mateixa pàgina es construirà un *iframe* del qual apareixeran el conjunt de les assignatures sobre les quals podem assignar-li el seu accés.

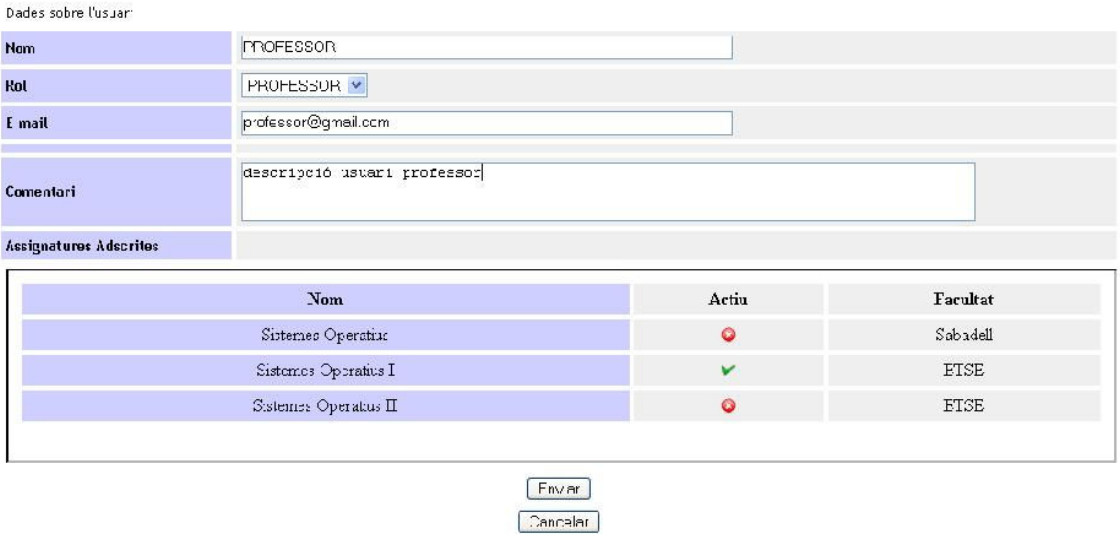


Figura : Eina administració usuaris.

4.9.3 Logs

Es definirà el sistema de logs com l'encarregat d'emmagatzemar els moviments més importants efectuats en el sistema. La construcció del sistema de logs es veu reflexada en dues taules del MySql, una serà la encarregada de definir les accions que els usuaris son capaços d'efectuar (*accions*) i la segona encarregada de contenir tots els moviments (*sys_log*)

Les següents accions seran les que inicialment es registraran com accions a la taula *accions:*

- Entrar al sistema
- Sortir del sistema
- Crear branca de temari

- Esborrar branca del temari
- Crear branca del temari
- Esborrar arxiu
- Esborrar pràctica
- Pujar Arxiu
- Modificar Nom Perfil
- Modificar email Perfil
- Crear assignatura
- Esborrar assignatura
- Crear usuari
- Activar assignatura
- Desactivar assignatura
- Modificar rol usuari
- Assignar assignatura a usuari
- Desassignar assignatura a usuari
- Crear pràctica
- Esborrar pràctica
- Esborrar Log
- Esborrar usuari
- Activar usuari
- Desactivar usuari

Cada acció tindrà un numero identificador que relacionarà en la taula *sys_log* la acció efectuada i sobre quin o quins elements s'han aplicat.

La pàgina encarregada del sistema de Logs tindrà les següents capacitats:

- **Cerca per accions** : Es disposarà d'un desplegable d'accions per a realitzar cerques i delimitar la cerca a una determinada acció (com per exemple esborrat de fitxers)

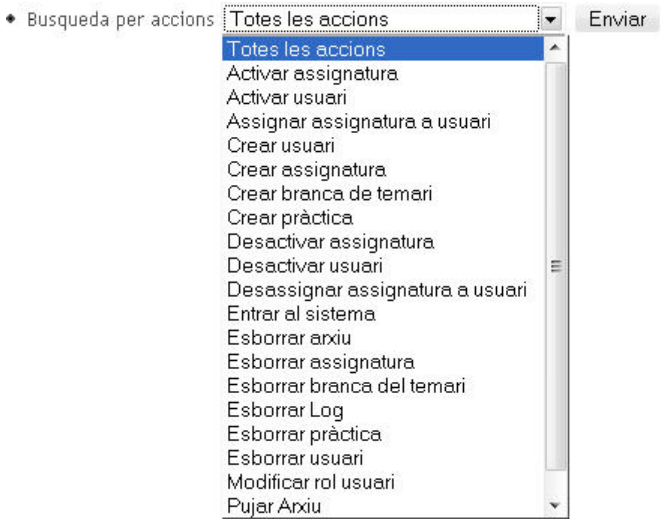


Figura : Formulari busqueda per accions

- **Cerca per dates** : La cerca limitada per dates acotarà les cerques de successos que hagin ocorregut en el sistema.

Figura : Formulari i calendari d'elecció de dates per la cerca.

- **Descarrega del registre de log en fitxer** : L'usuari amb rol administrador serà l'encarregat de conservar copies del sistema de logs per a possibles consultes posteriors, evitant així un tamany excessiu de la taula *sys_log*.

- **Esborrat del contingut actual de la taula *sys_log*:** La següent acció alliberarà tot el contingut emmagatzemat en la taula, serà recomanable realitzar una descarrega del fitxer abans d'efectuar un esborrat complet. Aquesta acció requerirà de confirmació i serà contemplada com la primera acció de la nova taula de registres.

- **Visionat del resultat de la recerca**: S'ofereix una vista preliminar del resultat de la recerca on s'indica la data, usuari, el rol que desenvolupa dins el sistema, l'acció juntament amb una descripció i el camp *Canvis* que indicaran diferents informacions segons les accions realitzades.

Data	Usuari	rol	Acció	Destinatari	Canvis
2007-06-07 21:35:52	ecesar	PROFESSOR	Esborrar arxiu	resolucio-prac-100000.swf	Sistemes Operatius I
2007-06-07 21:35:50	ecesar	PROFESSOR	Esborrar arxiu	suport-prac-100000.doc	Sistemes Operatius I
2007-06-07 21:35:48	ecesar	PROFESSOR	Esborrar arxiu	docresolucio-prac-100000.txt	Sistemes Operatius I
2007-06-07 21:35:45	ecesar	PROFESSOR	Esborrar arxiu	guio-prac-100000.pdf	Sistemes Operatius I
2007-06-07 21:35:43	ecesar	PROFESSOR	Esborrar arxiu	altres-prac-100000-1.rar	Sistemes Operatius I
2007-06-07 21:35:27	johan	ADMIN	Esborrar Log		

Figura : Vista de les accions efectuades en el sistema.

5. EL MINIKERNEL [CGP02]

Una de les parts en la que es centrarà el meu projecte recau en el Minikernel, un entorn simulat d'un sistema operatiu real, on aconseguirem desplaçar-nos de les habituals maneres d'entendre les pràctiques i on podrem abastar més àmpliament conceptes que abans quedaven sense ser contemplades en les sessions.

El ministerial ha estat desenvolupat per la Universitat Politècnica de Madrid sent la seva finalitat d'ús exclusivament didàctic.

Amb ell es busca que els alumnes aprenguin conceptes relacionats amb la gestió de processos i la multiprogramació, tocant directament el codi del operatiu simulat que ens ofereix aquesta base de treball. Buscar un entorn simulat facilita la comprensió i a la vegada la problemàtica de tocar directament el codi font del sistema operatiu, ja que aquest davant d'operacions no esperades podria deixar de funcionar amb les conseqüències que això comportaria.

La multiprogramació a la que ens referim té una importància rellevant ja que la concurrència és un eina molt important en el món dels sistemes operatius, l'alumne haurà de fer front a pràctiques en el que treballarà al nivell més baix del sistema operatiu simulat (Minikernel) i on experimentaran les mateixes dificultats a la que es veuen enfrontats els dissenyadors de sistemes operatius reals.

Em comentat a grans trets el que representava treballar sobre un entorn simulat, a continuació exposarem el perquè l'elecció d'aquesta plataforma davant d'altres que podrien semblar interessants però que com veurem presenten algunes deficiències :

- Els inconvenients que troba un alumne treballant directament sobre la màquina física comporta un llarga i complexa feina ja que no podrà recórrer a les eines habituals de depuració i es veurà obligat a un llarg cicle de reinicis cada vegada que un error en la programació faci que el sistema es col·lapsi.
- Avaluar una pràctica que ha estat desenvolupada directament sobre el sistema operatiu real comportaria una tasca dificultosa ja que l'instructor hauria de canviar el seu sistema per verificar cada una de les pràctiques, amb el risc que això comporta.
- El sistema MINIX podria ser més viable ja que no comporta tanta dificultat en la seva programació ja que va ser desenvolupat amb objectius únicament didàctics, però presenta els inconvenients que presentaven els sistemes operatius reals.
- Un altre sistema anomenat NACHOS, desenvolupat per la universitat de Berkeley, que tot i ser interessant presenta deficiències ja que utilitza un compilador creuat i del seu caràcter determinista.

Totes aquestes deficiències van dur a la Universitat Politècnica de Madrid a plantejar l'elaboració d'un sistema simulat (el Minikernel) que aportés més facilitat per ajudar a els seus alumnes la tasca de la programació i la comprensió del que comporta el disseny d'un sistema operatiu.

El Minikernel permet a nivell d'alumne tractar els següents conceptes relacionats amb la teoria impartida durant el curs :

- Arranc del sistema Operatiu
- Tractament d'interrupcions
- Tractament d'excepcions
- Tractament de crides al sistema
- La multiprogramació
- El canvi de context
- Planificació de processos
- Sincronització entre processos
- Tractament de l'Entrada / Sortida

L'objectiu vital per el que es va dissenyar fou que l'alumne aprengués el funcionament dels sistemes operatius dissenyant-los, només d'aquesta manera els continguts de les classes de teoria podien ser compresos i practicats.

Cal dir que juntament amb la comprensió del funcionament intern d'un sistema operatiu real (encara que sigui un entorn simulat), la programació de funcions orientades al Minikernel fa que l'alumne tingui més coneixement del llenguatge de C de programació, practicant sobretot el control de llistes i estructures.

5.1 Característiques del minikernel

La idea essencial del minikernel recau en el concepte de hardware virtual.

Els sistemes operatius reals ofereixen una capa de suport per al hardware creant una abstracció d'aquest oferint una màquina estesa sobre la que s'executen aplicacions.

El minikernel fa un enfoc diferent, el minikernel ens ofereix una capa per sobre del sistema operatiu (hardware virtual o sistema operatiu simulat) que crea una màquina real sobre de la màquina estesa.

Nivells de la màquina estesa més el minikernel :

- Programes d'usuari del minikernel
- Sistema operatiu del minikernel
- Hardware virtual (ofert per el minikernel)
- Sistema operatiu natiu
- Hardware real.

5.1.2 Característiques addicionals

- La capa del hardware virtual s'implementa fent servir els serveis POSIX del sistema operatiu natiu, evitant l'ús del codi ensamblador.

- No fa falta l'ús d'un compilador creuat, els programes dels usuaris com el codi del sistema operatiu es compilen utilitzant el compilador natiu de la màquina. Els

programes d'usuari son programes convencionals que fan crides a les funcions del sistema simulat en comptes de cridar les funcions de Linux/Unix.

- Els programes d'usuari com el codi del sistema operatiu s'executen sobre el processador de la màquina nativa, a diferencia del sistema NACHOS on els processos si que s'executen sobre el sistema simulat.

La versió actual del minikernel observarem que només disposa de dos sistemes d'entrada sortida : el rellotge i el terminal.

Figura representativa del sistema operatiu simulat sobre del sistema operatiu natiu

Podem observar que existeixen pocs dispositius d'entrada sortida però la Universitat Politècnica fa referència a aquest fet, mostrant els inconvenients d'afegir un sistema propi d'arxius o altres mòduls que farien del minikernel un sistema més complex deixant de banda el seu tret principal – senzillesa per a facilitar la comprensió al alumne.

L'entorn del minikernel s'ha mostrat com un sistema efectiu per a l'aprenentatge i desenvolupament de pràctiques vinculades a gestió de processos i d'entrada / sortida a baig nivell vinculats amb els dos dispositius senzills que disposa el minikernel : el rellotge i el terminal. Un altre aspecte important en el que va destinat és sobre el tractament de la multiprogramació dins el sistema operatiu, notant que a tan baix nivell tots els processos són asíncrons, mostrant les poques eines de depuració a aquest nivell i la gran dificultat per a la depuració donat el seu comportament no determinista.

L'alumne haurà d'anar trobant-se amb tots aquests problemes i adonar-se d'ells i enfrontar-se ja és tot un repte.

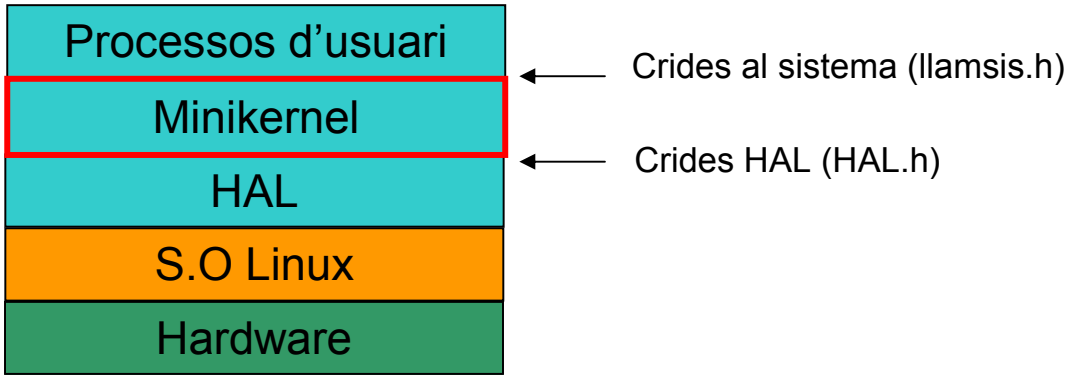
6. ENTORN DE LA PRÀCTICA

El Minikernel està dissenyat per a funcionar en qualsevol plataforma Linux, en el meu cas totes les proves seran realitzades en la versió 6,10 de Kubuntu.

6.1 Estructura del entorn de pràctiques

El Minikernel va ser concebut com un simulador d'un sistema real imitant el seu comportament i estructura. L'entorn consta de 3 parts vitals i que per al seu millor enteniment han estat dividits en subdirectoris :

- Directori BOOT – Cargador del sistema operatiu simulat
- Directori MINIKERNEL – Cos del sistema pròpiament dit.
- Directori USUARI – Programes que finalment l'usuari voldrà fer córrer sobre l'entorn simulat.



Com podem apreciar en el croquis del sistema, el minikernel conta amb una capa HAL aquest no es més que la capa de hardware virtual i que es convenient que els alumnes ho vegin com una caixa negre per no confondre i centrar-se en el codi del sistema operatiu.

Carrega del sistema operatiu

Com tots els sistemes operatius el minikernel ha de carregar el sistema operatiu en memòria i passar el control al seu punt inicial.

Per a realitzar la carrega del sistema utilitza un programa cargador, aquest es troba situat a la carpeta /*boot* i s'anomena de la mateixa manera *boot.*

Així una carrega normal del sistema simulat consistiria en la crida del nostre cargador indicant-li on es troba el nostre sistema operatiu, aquest últim situat a la carpeta */minikernel* amb nom *kernel.*

Comanda Carrega del minikernel:

boot/boot minikernel/kernel

El sistema operatiu simulat pararà la seva execució quan ja no tingui cap més programa d'usuari per executar.

6.2 El mòdul HAL

La capa HAL o de hardware virtual ofereix serveis per a que el sistema operatiu se'n faci càrrec.

A continuació descriuré les principals característiques:

- El processador pot prendre dos mode d'execució:
    1. Mode privilegiat on s'executa el codi del sistema operatiu
    2. Mode d'usuari on s'executen els processos d'usuari.

- El processador passarà de mode usuari a mode privilegiat sempre degut a algun tipus d'interrupció.

- Per a passar de mode privilegiat a mode usuari el sistema ha de modificar el valor de registre que indica l'estat del mode d'execució.

- El minikernel (com em comentat amb anterioritat) compta amb dos dispositius d'entrada/sortida basats en interrupcions: el terminal i el rellotge.

- El terminal es composa del teclat i la pantalla. La sortida de dades per pantalla es realitzen escrivint directament en la memòria del vídeo sense implicar l'ús d'interrupcions. El teclat per contra està dirigit per interrupcions que es produeixen cada cop que es prem una tecla.

- El rellotge té una freqüència d'interrupció programable.

- Existeix una taula de tractament d'interrupcions per la qual tractarem d'una manera o una altre aquestes.

Vector 0	Excepció aritmètica
Vector 1	Excepció per accés a memòria fallit
Vector 2	Interrupció de rellotge
Vector 3	Interrupció del terminal
Vector 4	Crida al sistema
Vector 5	Interrupció software.

Per tant podem veure que existeixen diferents nivells d'interrupció, tots ells no tenen la mateixa prioritat. Exposarem els diferents nivells de prioritats d'interrupcions.

Nivell 3 : Interrupció del rellotge
Nivell 2 : Interrupció del terminal

Nivell 1 : Interrupció software o crides al sistema
Nivell 0 : Execució en mode usuari

- El processador en cada moment executa en un determinat nivell d'interrupció i només admet interrupcions superiors al nivell actual
- Inicialment el processador s'executa en mode privilegiat i en el nivell més alt, per tant totes les interrupcions estaran inhibides.
- Al finalitzar la rutina d'interrupció s'ha de restaurar el nivell d'interrupció i el mode d'execució previ.
- El processador disposa de 6 registres de 32 bits de proposat general, que seran usats per el pas de paràmetres a crides del sistema.

A part de la simulació del aspecte hardware del sistema simulat, inclou funcions de gestió de memòria, fent que l'alumne no s'hagi de plantejar aspectes relacionats amb aquest tema i es centri en la gestió de processos i la entrada/sortida.

Operacions mòdul HAL :

- Operacions vinculades a la inicialització dels controladors de dispositius

void iniciar_cont_teclado( )
Inicialització del controlador de teclat

void iniciar_cont_reloj (TICKS_segon)
Inicialització del controlador del rellotge , com a paràmetre especificarem la freqüència d'interrupció desitjada en TICKS del sistema.

unsigned long long int leer_relog_CMOS( )
Utilitzant aquesta crida podrem llegir el registre de 64 bits podrem capturar la hora emmagatzemat pel rellotge del sistema.

- Operacions relacionades amb les interrupcions

void iniciar_cont_int( )
Inicialització del controlador d'interrupcions

void ubstak_man_int(int nvector, void (*controlador)( ))
Instal·lació d'un controlador d'interrupcions, en el vector corresponent
Tipus de controladors :

1. excepció aritmètica
2. excepció en accés a memòria
3. interrupció del rellotge
4. interrupció de terminal

Tipus de vectors :

1. vector de crides al sistema
2. vector per a les interrupcions software

int fijar_nivel_int(int nivel)
Aquesta funció és útil per a retornar al sistema el seu nivell original, retorna el nivell actual i permet fixar el sistema en el nivell que desitgem per permetre o inhibir totalment les interrupcions.

void activar_int_SW( )

Si el nivell d'interrupció ho permet aquesta funció provocarà una interrupció software.

int viene_de_modo_usuario ( )

La següent funció consulta el registre desat per la interrupció actual i permet conèixer si prèviament s'estava executant en mode usuari, retornant un valor cert en aquest cas.

- Operacions de gestió de la informació del context del procés

void fijar_contexto_ini (void *memoria, void *inicio_pila, int tam_pila, void *pc_inicial, contexto_t * contexto_ini);

Aquest servei crea el context inicial del procés, establint els valors inicials dels registres contador de programes, punter de l'adreça inicial de la pila i el seu tamany.

A més rep com a paràmetre una referència del mapa de memòria del procés i retornarà el context inicial segons els paràmetres atorgats en el paràmetre *contexto_ini*

void cambio_contexto (contexto_t *contexto_a_salvar, contexto_t *contexto_a_restaurar);

La rutina següent és l'encarregat de salvar el context d'un procés i restaurar al d'un altre (de gran utilitat quan haguem de bloquejar processos).

La restauració implica la copia del context rebut en els registres del processador.

Sinó s'especifica el context del procés a salvar, només es realitza la restauració.

- Funcions relacionades amb el mapa de memoria del procés

void *crear_imagen (char *prog, void **dir_ini);

Creem el mapa de memòria a partir del executable especificat *prog*.

Retorna un identificador del mapa de memòria creat, així com l'adreça d'inici del programa *dir_ini.*

void liberar_pila (void *mem);

Permet alliberar una imatge de memòria prèviament creat.

void *crear_pila (int tam);

Indicant el tamany de la pila reservarem un zona de memòria per a la regió de la pila, retornant com a resultat l'adreça inicial de la zona reservada.

void liberar_pila (void *pila);

Permet alliberar una pila prèviament creada

- Operacions varies amb funcions diverses.

int leer_registro (int nreg);
int escribir_registro (int nreg, int valor);

Aquestes dues rutines permeten llegir i escriure els registres de propòsit general del processador.

#define DIR_TERMINAL 1
char leer_puerto (int dir_puerto);

Funció que llegeix i retorna un byte del port d'Entrada/Sortida especificat.

En el nostre cas només disposarem del sistema de terminal.

void halt ( );

Atura la execució del processador fins que s'activi una interrupció.

void escribir_ker (char *buffer, unsigned int longi)

Aquesta funció permet escriure per pantalla les dades especificades en el paràmetre *buffer* , el paràmetre *longi* correspon amb el seu tamany.

int printk (const char *, ...);

Per a facilitar la utilització d'impressió per pantalla s'incorpora el *printk* que es comporta d'una manera similar al printf clàssic de C.

void panico (char *mens);

Dona la possibilitat d'aturar l'execució del sistema operatiu, mostrant el missatge especificat per pantalla.

6.3 El mòdul Kernel

Aquest mòdul conté el propi cos del nostre sistema operatiu simulat.

La primera versió i per tant la que s'entregarà al alumne representa un sistema monoprogramat, en el que encara que es puguin carregar en memòria varis programes, el procés en execució no deixarà els recursos fins que hagi acabat.

Durant el transcurs de les pràctiques l'alumne modificarà el codi permetent que aquest sigui un sistema multiprogramat.

Punts importants :

- Els processos són apropiatius, per tant un cop comencen a executar-se el procés captura els recursos i no els allibera fins a la seva finalització.
- L'estat d'un procés en execució serà en tot moment *en_ejecucion*.
- Només es cridarà a la rutina de context quan el procés acabi.

Caracteristiques principals de la versió inicial:

- **Estructura de Dades (** *kernel.h* **)**
  - *tabla_procs :* Taula de processos. Cada entrada representa un BCP. L'alumne podrà incloure nous camps quan ho consideri oportú.

- *lista_listos :* Cua de processos llests. Es tracta d'una llista amb enllaç simple. El tipus *lista_BCPs* guarda referències al primer i últim element de la llista.
- *p_proc_actual :* Apunta al BCP del procés en execució. Aquest BCP està inclòs a la cua de llests.
- *tabla_servicios :* Taula que guarda a cada posició l'adreça de la rutina del sistema operatiu que porta a terme la crida al sistema del qual el codi correspon amb aquesta posició.

- **Inicialització**

  Una vegada carregat el nostre sistema operatiu, el programa cargador passa el control al punt d'entrada d'aquest (a la funció *main* d'aquest mòdul). En aquest punt, el sistema inicia les seves estructures de dades i dispositius hardware i instal·la els seus controladors a la taula de vectors. Per últim m crea el procés inicial *init* i l'activa passant-li el control.

- **Tractament d'interrupcions externes.**

  Les úniques fonts externes d'interrupcions són el rellotge i el terminal.

  En el cas de les interrupcions per teclat, utilitza la funció *leer_puerto* per obtenir el caràcter del teclat.

- **Tractament d'interrupcions software.**

  De la mateixa manera que les rutines d'interrupció externes, només mostraran un simple missatge per pantalla.

- **Tractament d'excepcions.**

  Les dues possibles excepcions presents en el sistema tenen un tractament comú produint l'acabament del procés actual i la conseqüent alliberació dels seus recursos.

- **Crides al sistema**

  Existeix una única rutina d'interrupció per totes les crides. Tant el codi numèric com els paràmetres de la crida es passen a través de registres.

  Registre 0 : pas del codi

  Registre 1 cap endavant : pas dels paràmetres

  La crida comú al sistema obté el codi numèric de la crida i l'invoca indirectament a través de la *tabla_servicios* la funció corresponent a la crida.

  El minikernel inclou 3 crides en la seva versió inicial que passarem a descriure a continuació :

int crear_proceso(char *programa)

Crea un procés que executa el programa emmagatzemat en el fitxer especificat (en el paràmetre *programa*).

int terminar_proceso( )

Acaba l'execució d'un procés alliberant els seus recursos (BCP, mapa de memòria, pila, etc.)

int escribir(char *text, unsigned int longi)

Escriu un missatge per pantalla (paràmetre *text*) d'una determinada longitud (paràmetre *longi*) fent ús de la funció *escribir_ker* que varem comentar en el mòdul HAL

- **El planificador** (*funció planificador)*

  La versió inicial d'aquest mòdul consisteix en un sistema apropiatiu monoprogramat. Es basa en un algoritme de tipus FIFO que agafarà el primer element de la cua de llests canviant el seu estat per el de execució.

  En el suposat cas que tots els processos estiguessin a la cua de bloquejats, s'invoca a la rutina *espera_int* de la que no es torna fins que es produeix una interrupció.

6.4 Gestió de processos en el minikernel

La versió inicial d'aquest mòdul inclou una gestió bàsica de processos que correspon amb un sistema monoprogramat. Encara que en un principi puguem carregar en memòria múltiples programes, el procés romandrà en execució fins que acaba, ja sigui voluntàriament o no. A continuació es presenten les principals característiques del sistema operatiu inicial :

- La taula de processos és un vector de tamany fix de BCP.
- El BCP contindrà els següents camps:
  - Identificador del procés (id)
  - Estat (ACABAT, LLEST, EXECUCIÓ, BLOQUEJAT)
  - Copia dels registres de la CPU
  - Adreça inicial de la pila
  - Punter al següent BCP
  - Descriptor del mapa de memòria

Per al desenvolupament de les pràctiques que explicarem a continuació l'alumne haurà d'incorporar nous camps.

La variable *p_proc_actual* apunta al BCP del procés en execució i estarà inclosa dintre de la cua de llests.

Passos de la creació bàsica d'un procés

- Buscar entrada lliure a la memòria
- Llegir l'executable
- Crear mapa de memòria
- Reservar la pila del procés
- Crear el context inicial

- Omplir els camps corresponents de la BCP
- Posar el procés com a llest
- Insertar-lo a la llista de processos llests.

La alliberació d'un procés quan aquest ja ha acabat, implica alliberar els seus recursos, invocar el planificador per a que triï un altre procés i fer un canvi de context a aquest nou procés.

Cal remarcar un aspecte important en el fet que no existeix cap valor de procés nul i que si tots els processos existents estan bloquejats llavors l'últim en bloquejar-se queda executant-se la rutina *espera_int*, seguint executant aquest procés ja que la cua de llests roman buida. S'haurà de contemplar aquesta situació i decidir quan el sistema roman viu o ha entrat en un estat d'espera indefinida.

6.5 Els programes d'usuari

En el subdirectori *usuari* existeixen inicialment un conjunt de programes d'exemple que usen els serveis del minikernel. D'especial importància és el programa "init", ja que és el primer programa que arrenca el sistema operatiu. En un sistema real aquest programa consulta arxius de configuració per a arrencar altres programes que, per exemple, s'encarreguin d'atendre als usuaris De manera relativament similar, en el nostre sistema, aquest procés farà el paper de llançador d'altres processos, encara que en el nostre cas no es tracta de processos que atenguin a l'usuari, ja que el sistema no proporciona inicialment serveis per a llegir del terminal. Es tractarà simplement de programes que realitzen una determinada tasca i acaben.

Com passa en un sistema real, els programes tenen accés a les crides al sistema com rutines de biblioteca. Per a això, existeix una biblioteca estàtica, denominada "libserv.a", que conté les funcions d'interfície per a les crides al sistema.

Els programes d'usuari no han de fer servir crides al sistema operatiu natiu, encara que si podran fer servir funcions de la biblioteca estàndard de C, com, per exemple, "strcpy" o "memcpy".

La biblioteca "libserv.a" està emmagatzemada en el subdirectori "usuari/lib" i està composta de dos mòduls:

"serv". Conté les rutines d'interfície per a les crides. Es recolza en una funció del mòdul "misc" denominada "llamsis", que és la qual realment executa la instrucció de cridada al sistema. Per a fer accessible als programes una nova cridada, l'alumne haurà de modificar aquest arxiu per a incloure la rutina d'interfície corresponent.

"misc". Com intenta indicar el seu nom, aquest mòdul conté un conjunt divers de funcions d'utilitat. Entre elles, la definició de la funció "printf" que, evidentment, es recolza en la cridada al sistema "escriure", de la mateixa manera que el "printf" en un sistema UNIX es recolza en la crida "write". Així mateix, proporciona la funció de conveniència "llamsis". Aquesta funció facilita la invocació d'una crida al sistema ocupant-se de la tediosa tasca d'emplenar els registres amb els valors.

7. PRÀCTIQUES AMB EL MINIKERNEL [ARCOS]

A continuació i fent us del material subministrat per la Universitat Politècnica de Madrid destruiré els enunciats i possibles solucions als problemes plantejats.

ASSIGNATURA: SISTEMES OPERATIUS.

7.1.1 INCLUSIÓ D'UNA CRIDA SIMPLE

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, desenvolupant un dels passos vitals per a la definició de noves crides al sistema.

Comprendre l'organització del entorn i familiaritzar-se amb el nou entorn de pràctiques.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.

S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

En aquest primer exercici sobre el minikernel s'haurà de modificar la versió inicial del minikernel (entregada com a material de suport) per a incloure noves funcionalitats.

Sempre que l'alumne compleixi amb les especificacions requerides aquest tindrà total llibertat sobre el disseny i programació del que s'està demanant.

La versió inicial disposa de tres crides al sistema que us serviran de referència :

- CREAR_PROCESO
- TERMINAR_PROCESO
- ESCRIBIR

FUNCIONAMENT

A continuació indicarem els passos típics per a la inclusió de noves crides al sistema per a facilitar la realització d'aquesta primera pràctica.

Suposant que la nova crida s'anomena *nova_crida* els passos a realitzar seran els

següents:

- Incloure en ***minikernel/kernel.c*** el codi de la nova rutina, en el nostre cas afegirem sis_nova_crida.
- Incloure a la taula de serveis que trobareu al fitxer ***kernel/include/kernel.h*** la nova crida al sistema afegint-la a la última posició de la taula.

- Modificar el fitxer ***kernel/include/llamsis.h*** incrementant en 1 el numero de les crides a sistema disponibles i assignar el codi més alt a la nova crida.
- Per finalitzar, en aquest punt el nostre sistema operatiu simulat ja disposaria de la nova crida al sistema, però només seria accessible usant codi ensamblador. Per això modificarem les biblioteques **usuario/lib/serv.c** perquè ens ofereixi una interfície amb el nou sistema, de la mateixa manera modificarem el fitxer ***usuario/include/servicios.h*** per a que disposin del prototip de la funció d'interfície.

PART 1 – Obtenció identificador de procés

S'ha d'afegir la següent crida al sistema:

int get_id_proces ( );

El seu funcionament és senzill i només haurà de retornar l'identificador del procés que l'ha invocat.

Nota : per a poder treure el numero identificador haureu de consultar el camp oportú de la estructura BCP lligada al procés.

PART2 – Obtenció del temps de vida d'un procés

De la mateixa manera inclourem :

int get_iemps_vida_proces ( );

Ens retornarà el temps de vida d'un procés qualsevol en TICKS del rellotge, per a poder-ho realitzar haurem d'introduir nous camps a la estructura BCP per a poder obtenir aquesta informació.

Nota : es recomana utilitzar una variable global amb el numero de TICKS que han transcorregut fins al moment de forma que des de que s'inicialitza el minikernel aquest comença des de 0 i va incrementant en 1 TICK a cada interrupció del rellotge.

7.1.2 Comentaris i resolució:

PART 1 – Obtenció identificador de procés

La tasca de la pràctica és incloure noves cridades al sistema, s'ha considerat oportú incloure en aquesta secció els passos típics que cal portar a terme en aquest sistema per a fer-lo. Suposant que el nou servei es denomina "nova", aquests són els passos a realitzar:

- Incloure en "minikernel/kernel.c" una rutina (que podria denominar-se "sis_nova") amb el codi de la nova cridada.

- Incloure en "tabla_servicios" (fitxer "minikernel/include/kernel.h") la nova crida en l'ultima posició de la taula.

- Modificar el fitxer "minikernel/include/llamsis.h" per a incrementar el nombre de cridades disponibles i assignar el codi més alt a la nova cridada.

- Una vegada realitzats els passos anteriors, el sistema operatiu ja inclouria el nou servei, però només seria accessible des dels programes usant codi ensamblador. Per tant, és necessari modificar la biblioteca de serveis (fitxer "usuari/lib/serv.c") perquè proporcioni la interfície per al nou servei. S'hauria de també modificar el fitxer de capçalera que inclouen els programes d'usuari ("usuari/include/serveis.h") perquè disposin del prototip de la funció d'interfície.

- Finalment, cal crear programes de prova per a aquest nou servei i, evidentment, modificar "init" perquè els invoqui. Així mateix, s'hauria de modificar el fitxer "Makefile" para facilitar la compilació d'aquest nou programa.
Per a poder resoldre aquest apartat narraré pas a pas el que l'alumne hauria o ha de fer per a la seva correcte resolució.

- Pas1: Incloure en ***minikernel/kernel.c*** el codi de la nova rutina

Rutina d'obtenció del id del procés actual

```
int sis_obtenir_id ( )
{
return p_proc_actual->id;
}
```

L'explicació és molt senzilla p_proc_actual és del tipus de l'estructura BCP per tant si volem obtenir el seu identificador i sabent que l'estructura BCP ens la pot proporcionar només em de retornar el camp identificador.

L'alumne haurà de veure que aquesta variable està creada i comentada la seva utilització en el fitxer ***minikernel/include/kernel.h***

- Pas2: Incloure a la taula de serveis que trobareu al fitxer ***kernel/include/kernel.h*** la nova crida al sistema afegint-la a la última posició de la taula.

Afegirem a els prototips de les rutines que realitzen crides al sistema la nova crida.

```
int sis_obtenir_id ( ) ;
```

i actualitzarem la taula de serveis oferint la nova crida

```
servicio tabla_servicios[NSERVICIOS] =
{
{sis_crear_proceso},
{sis_terminar_proceso},
{sis_escribir},
{sis_obtenir_id}
};
```

- Pas3: Modificar el fitxer ***kernel/include/llamsis.h***

Incrementarem el numero de serveis oferts del nostre kernel augmentant en 1 la variable NSERVICIOS i la situarem en 4 ja que com em comentat anteriorment la versió inicial del minikernel només ens permetrà 3 crides al sistema diferents.

```
#define NSERVICIOS 4
```

I assignarem el codi més alt a la nova crida

```
#define OBTENIR_ID
```

En aquest moment el servei d'obtenció de l'identificador d'un procés ja és operatiu ara tot seguit dotarem d'una interfície per a poder ser utilitzades per les aplicacions dels usuaris:

- Pas4: Modificació de la biblioteca de recursos **usuario/lib/serv.c**

Com em denotat anteriorment afegirem la nova funció interface per a les crides al sistema, sent aquesta :

```
int obtenir_id (){
```

```
return llamsis(OBTENIR_ID, 0);
}
```

Aquesta funció és la que realment realitza la crida al sistema, *llamsis* ens indicarà primer quina crida volem realitzar i el segon paràmetre quants arguments volem passar-li.

- Pas5: Modificació de les capçaleres dels serveis ***usuario/include/servicios.h***

  Afegirem la crida al sistema proporcionats al usuari

  *Incloure en **minikernel/kernel.**c el codi de la nova rutina, en el nostre cas afegirem sis_nova_crida.int obtenir_id ( );*

  Un cop modificat el fitxer Makefile i init.c el primer per compilar el nou fitxer i el segon per a que el Sistema Operatiu executi el programa del alumne amb la crida crear_proceso ('nom executable'), ja només caldrà que l'alumne realitzi un fitxer que executi la nova crida al sistema.

Exemple programa usuari i resposta:

Fitxer yosoy.c

```
#include "servicios.h"
#define TOT_ITER 10
int main()
{
    int i, id;
    id=obtener_id_pr();
    for (i=0; i<TOT_ITER; i++)
        printf("yosoy (%d): i %d\n", id, i);
    printf("yosoy (%d): termina\n", id);
    return 0;
}
```

*Crearem dins de **usuario/init.c** dos processos i observarem que tenen identificadors diferents.*

Sortida del programa

```
init: comienza
-> PROC 0: CREAR PROCESO
-> PROC 0: CREAR PROCESO
init: termina
-> FIN PROCESO 0
-> C.CONTEXTO POR FIN: de 0 a 1
yosoy (1): i 0
yosoy (1): i 1
yosoy (1): i 2
yosoy (1): i 3
yosoy (1): i 4
yosoy (1): i 5
yosoy (1): i 6
yosoy (1): i 7
yosoy (1): i 8
yosoy (1): i 9
yosoy (1): termina
-> FIN PROCESO 1
-> C.CONTEXTO POR FIN: de 1 a 2
yosoy (2): i 0
yosoy (2): i 1
yosoy (2): i 2
yosoy (2): i 3
yosoy (2): i 4
yosoy (2): i 5
yosoy (2): i 6
yosoy (2): i 7
yosoy (2): i 8
yosoy (2): i 9
yosoy (2): termina
-> FIN PROCESO 2
```

PART2 – Obtenció del temps de vida d'un procés

L'alumne ha d'entendre que el que s'està sol·licitant és una funció que retorni el temps de vida d'un procés en el numero bàsic de TICKS en la que la interrupció del rellotge del nostre sistema simulat s'executa.

Per tant s'haurà de crear una simple funció que retorni el temps actual del sistema, d'aquesta manera sabent quan ha començat i quan ha acabat sabrem quin ha estat el temps de vida del nostre procés, a continuació descriurem el funcionament i les inclusions de codi que s'han d'afegir a la base del minikernel per afegir aquesta nova funcionalitat.

- Pas1: Incloure en ***minikernel/kernel.c*** el codi de la nova rutina

  Primer de tot s'ha de definir una variable global que conti el numero de TICKS que el sistema porta consumits, per tant dins l'arxiu ***minikernel/include/kernel.h*** definirem inicialitzarem el contador *num_TICKS a 0.*

  Ara ja només caldrà anar incrementant en cada interrupció del rellotge aquesta variable en una unitat (afegir l'increment dins la funció *int_reloj)* i fer una funció que retorni el temps actual (de TICKS) del sistema

  Rutina d'obtenció del temps actual

```
int sis_get_temps_actual ( )
{
return num_TICKS;
}
```

- Pas2: Incloure a la taula de serveis que trobareu al fitxer ***kernel/include/kernel.h*** la nova crida al sistema afegint-la a la última posició de la taula.

Afegirem a els prototips de les rutines que realitzen crides al sistema la nova crida.

```
int sis_get_temps_actual ( );
```

i actualitzarem la taula de serveis oferint la nova crida

```
servicio tabla_servicios[NSERVICIOS]=
{
{sis_crear_proceso},
{sis_terminar_proceso},
{sis_escribir},
{sis_coger_id_proceso},
{sis_get_temps_actual ( )}
};
```

- Pas3: Modificar el fitxer ***kernel/include/llamsis.h***

Incrementarem el numero de serveis oferts del nostre kernel augmentant en 1 la variable NSERVICIOS.

```
#define NSERVICIOS 5
```

I assignarem el codi més alt a la nova crida

```
#define GET_TEMPS 4
```

- Pas4: Modificació de la biblioteca de recursos **usuario/lib/serv.c**

```
int get_temps( ){
      return llamsis(GET_TEMPS, 0);
}
```

- Pas5: Modificació de les capçaleres dels serveis ***usuario/include/servicios.h***

Afegirem la crida al sistema proporcionats al usuari

```
int get_temps ( );
```

En el programa usuari només caldrà capturar el temps en el inici i al final de la execució del procés i obtindrem el temps de vida en TICKS del procés actual.

Fitxer yosoy.c

```
#include "servicios.h"
#define TOT_ITER 500
int main()
{
      int i, id,temps1,temps2,temps_de_vida;
     temps_de_vida=0;
     temps1=get_temps();
      id=obtener_id_pr();
      for (i=0; i<TOT_ITER; i++)
            printf("yosoy (%d): i %d\n", id, i);
       printf("yosoy (%d): termina\n", id);
       temps2=get_temps();
       temps_de_vida=temps2-temps1;
       printf("Temps de vida (%d) TICKS: inici (%d) : final (%d)\n",
            temps_de_vida,temps1,temps2);
      return 0;
}
```

Sortida del programa

```
init: comienza
-> PROC 0: CREAR PROCESO
-> PROC 0: CREAR PROCESO
init: termina
-> FIN PROCESO 0
-> C.CONTEXTO POR FIN: de 0 a 1
yosoy (1): i 0
yosoy (1): i 1
yosoy (1): i 2
...
yosoy (1): i 484
yosoy (1): i 485
-> TRATANDO INT. DE RELOJ
yosoy (1): i 486
yosoy (1): i 487
...
yosoy (1): i 498
yosoy (1): i 499
yosoy (1): termina
Temps de vida (1) TICKS: inici (0) : final (1)
-> FIN PROCESO 1
```

```
-> C.CONTEXTO POR FIN: de 1 a 2
yosoy (2): i 0
yosoy (2): i 1
yosoy (2): i 2
...
yosoy (2): i 155
yosoy (2): i 156
-> TRATANDO INT. DE RELOJ
yosoy (2): i 157
yosoy (2): i 158
...
yosoy (2): i 435
yosoy (2): i 436
-> TRATANDO INT. DE RELOJ
yosoy (2): i 437
yosoy (2): i 438
...
yosoy (2): i 498
yosoy (2): i 499
yosoy (2): termina
Temps de vida (2) TICKS: inici (1) : final (3)
-> FIN PROCESO 2
```

ASSIGNATURA: SISTEMES OPERATIUS.

7.2.1 BLOQUEIG DE PROCESSOS

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, desenvolupant un dels passos vitals per a la definició de noves crides al sistema.

Aprofundir en la comprensió de la estructura interna del minikernel.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.

S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

Per a la realització de la següent pràctica caldrà afegir una nova crida al sistema que bloquejarà durant un temps donat un procés.

El plaç de temps s'especificarà en segons dins de la crida al sistema.

int dormir (unsigned int segons)

Noteu que al afegir processos bloquejats estarem afegint multiprogramació al nostre sistema, ja que quan un procés cridi aquesta funció el planificador assignarà l'execució a un altre procés.

Per a la facilitació de la realització de la pràctica a continuació indicarem els passos a seguir per a la seva correcte resolució, recordeu que cada alumne pot resoldre la pràctica de la manera que li sembli convenient.

- Incloure el camp a la BCP relacionat amb el plaç de la crida dormir.
- Definir una llista de processos bloquejats o dormits durant un cert període de temps.

- Incloure la nova crida

  int sis_dormir( )

  Aquesta haurà de posar el procés a la llista de bloquejats, per facilitar la seva elaboració caldrà crear les següents funcions :

  static void bloquejar(llista_BCPs *lista) { }

  S'encarregarà de posar el procés en la llista de bloquejats.

  Algoritme :

  Canviar l'estat del procés actual canviant BCP i indicant el seu nou estat

  Copiar les dades de la BCP del procés actual

  Fixar el nivell d'interrupcions en el 3 (Interrupcions del rellotge, guardant a la variable "nivell", el nivell anterior).

  Treure de la llista de processos llests el procés a bloquejar

  Insertar el procés bloquejat a la nova llista de processos bloquejats

  Cridar al planificador per emmagatzemar el següent procés següent a executar.

  Canviar el context del processador, salvant el context del procés bloquejat i restaurant els registres del procés següent al processador.

  static void desbloquear(BCP * proc, lista_BCPs *lista) { }

  Per a facilitar la reutilització, el procés que sigui passat per paràmetre serà desbloquejat de la llista que li especifiquem, en el nostre cas la llista de bloquejats.

  Algoritme :

  Canviar l'estat del procés a la BCP

  Guardar nivell anterior i fixar el nivell inhibint interrupcions.

  Eliminar el procés de la llista de bloquejats

  Incloure procés a la llista de llests.

  Restaurar el nivell anterior

- Afegir a la rutina d'interrupció del rellotge la detecció de venciment del plaç que un procés ha estat dormit. Un cop aquest procés hagi vençut el plaç caldrà posar-lo a la llista de processos llests, traient-lo de la llista de bloquejats.

Comproveu que dos processos que es dormen simultàniament (dintre del mateix TICK del sistema) al despertar-se ho hauran de fer durant la mateixa interrupció de rellotge.

Prova del servei dormir

Es tracta d'un programa ("provadormir") que llança l'execució de dos processos que executen el mateix programa ("dormilon"). Aquest programa invoca dues vegades al servei "dormir":

- La primera amb un valor d'un segon
- La segona amb un valor que depèn del seu pid ("segs=pid+1").

S'han de comprovar els següents aspectes:

- Que els processos dormen el nombre de ticks apropiat (100 per cada segon).
- Que la primera vegada que es dormen els dos processos "dormilon" es desperten simultàniament amb la mateixa interrupció de rellotge.

7.2.2 Comentaris i resolució

La segona pràctica intenta involucrar més a l'alumne en canvis més profunds i en experimentar amb els canvis de context.

S'ha d'incloure una nova cridada ("int dormir(unsigned int segons)") que permeti que un procés pugui quedar-se bloquejat un termini de temps. El termini s'especifica en segons com paràmetre de la crida. La inclusió d'aquesta crida significarà que el sistema passa a ser multiprogramat, ja que quan un procés l'invoca passa a l'estat bloquejat durant el termini especificat i s'haurà d'assignar el processador al procés triat pel planificador. Notem que en el sistema només existiran canvis de context voluntaris i, per tant, segueix sense ser possible l'existència de crides al sistema concurrents. No obstant això, la rutina d'interrupció del rellotge va a manipular llistes de BCPs, és necessari revisar el codi del sistema per a detectar possibles problemes de sincronització en el maneig d'aquestes llistes i solucionar-los elevant el nivell d'interrupció en els fragments de codi corresponents. Encara que l'alumne pugui implementar aquesta crida com consideri oportú, a continuació se suggereixen algunes pautes:

- Modificar el BCP per a incloure algun camp relacionat amb aquesta cridada.
- Definir una llista de processos esperant terminis.
- Incloure la crida que, entre altres tasques, ha de posar al procés en estat bloquejat, reajustar les llistes de BCPs corresponents i realitzar el canvi de context.
- Afegir a la rutina d'interrupció la detecció de si es compleix el termini d'algun procés dormit. Si és així, ha de canviar-li d'estat i reajustar les llistes corresponents.
- Revisar el codi del sistema per a detectar possibles problemes de sincronització i solucionar-los adequadament.

L'alumne ha de veure l'estructura bàsica i el funcionament de les diferents llistes i modes en el que els processos es troben.

- Incloure el camp a la BCP relacionat amb el plaç de la crida dormir. (kernel.h)

```
typedef struct BCP_t {
    int id;
    int estado;
    contexto_t contexto_regs;
    void * pila;
    BCPptr siguiente;
    void *info_mem;
    unsigned int plazo;          /* ticks que falten per despertar-se */
} BCP;
```

- Definir una llista de processos bloquejats o dormits durant un cert període de temps. (kernel.h)

```
/*
 * Variable global que representa la cua de processos dormits
 */
lista_BCPs lista_dormidos= {NULL, NULL};
```

- Inclusió nova crida a la taula de serveis(kernel.h) i incrementació del numero de serveis oferts (llamsis.h)
- Modificació de la biblioteca de serveis (serv.c)

```
int dormir(unsigned int segundos)
{     return llamsis(DORMIR, 1, (long)segundos);  }
```

- Incloure en ***minikernel/kernel.c*** el codi de la nova rutina

  Per a poder dormir un procés aquest haurà de canviar de pila, deixar d'estar en mode llest per a la execució i situar-se a la nova llista que crearem de bloquejats.

  Per tant haurem de definir dos noves rutines una per bloquejar i una altre per desbloquejar processos.

  Funció per a bloquejar processos :

```
/*
 * Funció que bloqueja el procés que està actualment en execució
 */
static void bloquejar(llista_BCPs *lista) {
      BCP * p_proc_anterior;
      int nivell;
      p_proc_actual->estado=BLOQUEADO;
     //canviar l'estat del procés actual canviant BCP i indicant el seu nou estat
      p_proc_anterior=p_proc_actual;
      //copiem les dades  de la BCP del procés actual
      nivell=fijar_nivel_int(NIVEL_3);
      //fixem el nivell d'interrupcions en el 3 (Interrupcions del rellotge, guardant a la variable "nivell", el nivell anterior.
      eliminar_primero(&lista_listos);
      // traiem de la llista de processos llests el procés a bloquejar
      insertar_ultimo(lista, p_proc_anterior);
     // insertem el procés bloquejat a la nova llista de processos bloquejats
      p_proc_actual=planificador();
```

```
        //cridem al planificador perquè ens retorni el següent procés.
         cambio_contexto(&(p_proc_anterior->contexto_regs),
                                &(p_proc_actual->contexto_regs));
        //canvi context del procés anterior amb l'actual.
         fijar_nivel_int(nivell);
        //restaurem el nivell d'interrupcions anterior
    }
```

Funció per a desbloquejar processos:

```
/*
 * Funció que desbloqueja un procés
 */
static void desbloquear(BCP * proc, lista_BCPs *lista) {
        int nivel;

        proc->estado=LISTO;
        //canviar l'estat del procés a la BCP
        nivel=fijar_nivel_int(NIVEL_3);
        //guardar nivell anterior i fixar el nivell inhibint interrupcions.
        eliminar_elem(lista, proc);
        //eliminem el procés de la llista corresponent
        insertar_ultimo(&lista_listos, proc);
        //tornar a possar el procés a la llista de llests
        fijar_nivel_int(nivel);
        //restaurem el nivell anterior

}
```

Inclusió de la nova crida

```
/*
 * Tractament de la crida dormir
 */
int sis_dormir(){
        unsigned int segundos;

        segundos=(unsigned int)leer_registro(1);
        /* Establim quants TICKS de rellotge em d'esperar llegint el registre 1,
        ja que només se li ha passat una variable per paràmetre */
        p_proc_actual->plazo=segundos*TICK;
        /* En el procés actual se li actualitza el plaç d'espera en la taula BCP */
        bloquear(&lista_dormidos);
        /* Cridem a la funció bloquejar per a "dormir" el procés actual */
        return 0; /* mai retornarà error */
}
```

Per últim només ens falta incloure una rutina dins la rutina d'interrupció del rellotge que vagi actualitzant el camp de plaç pendent " a dormir " de cada procés a la BCP.

```
/*
 * Tractament de les interrupcions del rellotge
 */
static void int_reloj(){

        printk("-> TRATANDO INT. DE RELOJ\n");
        ajustar_dormidos();

      return;
}
//
```

```
/*
 * Funció auxiliar que decrementa el plaç d'espera dels processos
 * dormits. Si algun arriba a 0 ,es desbloqueja
 */
static void ajustar_dormidos() {
        BCP * p_aux;
        BCP * p_sig;

        p_aux=lista_dormidos.primero;
        /* Agafem el primer procés de la llista de dormits */
        /* BUCLE : Per a cada procés que precedeixi al procés actual si després de
        decrementar-li en una unitat el seu plaç d'espera aquest és igual a 0 cridarem a
        la funció desbloquejar. */
        while (p_aux) {
                p_sig=p_aux->siguiente;
                if (--p_aux->plazo==0)
                        desbloquear(p_aux, &lista_dormidos);
                p_aux=p_sig;
        }
}
```

Per últim només caldrà dissenyar un arxiu d'usuari fent les crides corresponents a les noves funcions incorporades al sistema.

Fitxer Usuari **dormilon.c**

```
#include "servicios.h"

int main(){
        int segs, id;

        id=obtener_id_pr();
        printf("dormilon (%d): comienza\n", id);

        /* primer dorm 1 segon */
        printf("dormilon (%d) duerme 1 segundo\n", id);
        dormir(1);

        /* després dorm el nombre de segons depenen del seu pid */
        segs=id+1;
        printf("dormilon (%d) duerme %d segundos\n", id, segs);
        dormir(segs);

        printf("dormilon (%d): termina\n", id);
        return 0;
}
```

ASSIGNATURA: SISTEMES OPERATIUS.

7.3.1 PLANIFICACIÓ ROUND-ROBIN

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, afegint funcionalitats relacionades amb la planificació.
Aprofundir en la comprensió de la estructura interna del minikernel.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.
S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

La versió inicial del sistema operatiu presenta un algoritme FIFO que haurà de ser substituït per un algoritme *round-robin*..

Recordatori :

Round robin es un metode per seleccionar tots els elements d un grup de manera equitativa i en un ordre racional, normalmente començant pel primer element de la lista fins arribar al últim i tornant a començar desde el primer element.

Una forma senzilla de entendre el round robin és imaginar una sequencia per "agafar torns". En operacions computacionals, un metode per executar diferents processos de manera concurrent, per a la utilització equitativa dels recursos del equip, es limitant cada proces a un petit periode de temp (quantum), i despres suspenent aquest proces per donar oportunitat a un altre proces i així sucesivament.
- wikipedia –

El minikernel disposa de variables de gran utilitat que poden ser trobades dins el fitxer *const.h*
El tamany del quantum estarà definida per la variable TICKS_POR_RODAJA, inicialment atorgarà 10 TICKS de rellotge per cada procés.

Per tant, cada vegada que passin 10 TICKS de rellotge, el nostre sistema haurà de cridar al planificador per a que decideixi el següent procés a executar canviant el context, si és necessari.

Si el quantum es consumís quan aquest es troba en **mode sistema**, per evitar errors, no s'ha de realitzar el canvi de context, i aquest canvi queda pendent fins al **mode usuari.**

Haureu d'incloure dins la rutina d'interrupció de software aquest canvi de context vigilant si hi ha hagut o no un canvi pendent que s'hagi produït dins del mode sistema.

Prova del planificador round robin

Hi Haurà dues proves independents: una en la qual els processos estan contínuament cridant a "printf" (amb el que en la majoria dels casos la interrupció de rellotge que indica la fi de rodanxa "enxamparà" al procés fent una cridada al sistema) i altra en la qual els processos no estaran fent cridades al sistema.

En ambdues proves es creen 5 processos i el resultat ha de ser tal que els processos es reparteixin el temps proporcionalment entre ells, o sigui, que en la sortida generada per la prova, els 5 processos han d'acabar en la fase final d'aquesta sortida.

En la primera prova ("prova*RR/1") els processos creats executen el programa "yosoy" que mostra contínuament la seva identitat per la pantalla.

Execució procés 1 **Execució procés 1** **...** **Execució procés 1** **Execució procés 2** **Execució procés 2** **...** **Execució procés 2** **Execució procés 1** **Execució procés 1** **...** **Execució procés 1**	**Execució procés 2** **Execució procés 2** **...** **Execució procés 2**

En la segona prova (”prova*RR/2”) els processos creats executen el programa “mut” que no escriu per la pantalla però realitza càlculs aritmètics per a “gastar processador”.

A més de comprovar que les proves del round-robin funcionen correctament, asseguri’s que el seu programa tracta adequadament les dues següents situacions:

- Quan un procés es desbloqueja i passa a executar se li assignarà una rodanxa completa.

- Si es compleix la rodanxa mentre un procés està fent una cridada al sistema (p. ex. “*lock”), s’activa que hi ha un canvi de context involuntari pendent, però si quan el procés continua amb la cridada es queda bloquejat, s’haurà de desactivar el canvi pendent per a no repercutir-lo a altre procés.

7.3.2 Comentaris i resolució

La següent pràctica intentarà involucrar al alumne en el sistema planificador del sistema.

Es va a substituir l'algorisme de planificació FIFO per round robin, on la grandària de la rodanxa serà igual a la constant "TICKS_POR_RODANJA". Amb la inclusió d'aquest algorisme, apareixen canvis de context involuntaris, el que causa un gran impacte sobre els problemes de sincronització dintre del sistema al poder-se executar diverses crides de forma concorrent. Per a solucionar aquests problemes, en la pràctica no es van a permetre els canvis de context involuntaris mentre el procés està executant la rutina de tractament d'un dispositiu o una crida al sistema (es tracta d'un nucli no expulsiu). Per a assolir aquest objectiu, la solució plantejada es va a basar en el mecanisme d'interrupció programari. Així, en la rutina de tractament de la interrupció programari es realitzarà el canvi de context del procés actual passant-lo al final de la cua de llests. A més, la implementació del round robin ha de cobrir els següents aspectes:

- A assignar el processador a un procés, se li ha de concedir sempre una rodanxa completa, amb independència de si la rodanxa prèvia la ha consumit completament o no.
- Si un procés que té pendent un canvi de context involuntari es bloqueja com part de l'execució d'una cridada, no ha d'aplicar-se aquest canvi.

En la teoria ja hauran estudiat tots els mecanismes, així doncs que aquesta pràctica servirà per observar els canvis que es produeixen en un sistema "real" tan en execució com en canvis a realitzar en codi.

Modificacions sobre ***minikernel/kernel.h***

- Canvis en la estructura BCP indicant el temps de la "rodanxa" d'apropiació (quàntum)

```
typedef struct BCP_t {
    int id;
    int estado;
    contexto_t contexto_regs;
    void * pila;
    BCPptr siguiente;
    void *info_mem;
    unsigned int plazo;
    unsigned int rodaja;          /* quan queda del quàntum actual */
} BCP;
```

- Inclusió d'una variable global indicant canvis pendents de context, ja que quan un quàntum acaba dins del mode servei aquest ha de quedar marcat com a pendent.

```
/*
* Variable global que indica si hi ha una replanificació pendent,
 * es a dir, un canvi de context involuntari pendent per aplicar-se
 *
 */
int replanificacion_pendiente=0;
```

Modificacions sobre ***minikernel/kernel.c***

- Definició de la funció encarregada de actualitzar el temps de vida d'un procés durant la seva execució.

```
/*
 * Funció auxiliar que actualitza la rodanxa i si detecta el seu acabament
 * activa una interrupció software.
 */
static void actualizar_rodaja() {
 /* Si el procés no esta llest per executar-se, no s'actualitza la
    rodanxa. */
 if (p_proc_actual->estado == LISTO) { /* si el procés està llest */
       p_proc_actual->rodaja--; /* disminuïm el seu temps de vida dins el quàntum */
       if (p_proc_actual->rodaja==0) { /* si el temps de vida s'exhaureix */
                replanificacion_pendiente=1; /* anotem un canvi pendent */
                activar_int_SW(); /* activem la interrupció Software */
       }
 }
}
```

- Inclusió del tractament de replanificació dins la rutina d'interrupció de software.

```
/*
 * Tratamiento de interrupciones software
 */
static void int_sw(){

        printk("-> TRATANDO INT. SW\n");
        /* Hi ha una replanificació pendent? */
        if (replanificacion_pendiente)
                cambio_proceso(&lista_listos);

        return;
}
```

- Modificació de la creació de processos atorgant el temps de vida a cada procés

```
static int crear_tarea(char *prog){

.....
                /* Inicialment s'assigna una rodanxa completa */
                p_proc->rodaja=TICKS_POR_RODAJA;

....
}
```

- Inclusió dins de la interrupció de rellotge de la crida a la funció de decrement del temps de vida del procés dins del quàntum.

```
/*
 * Tractament d'interrupcions del rellotge
 */
static void int_reloj(){
        printk("-> TRATANDO INT. DE RELOJ\n");
        actualizar_rodaja();
        ajustar_dormidos();

  return;
}
```

- Caldrà modificar la funció “bloquejar” de la pràctica anterior per a que realitzi el comportament bàsic de la planificació round-robin.

```
/*
 * Funció que realitza un canvi de procés ja sigui voluntari o involuntari,
 * incloent el canvi context voluntari vinculat a l'acabament d'un
 * procés ( lista_destino==NULL i p_proc_actual->estado==TERMINADO)
 */
static void cambio_proceso(lista_BCPs *lista_destino) {
        BCP * p_proc_anterior;
        contexto_t *contexto_aux;
        int nivel;

        p_proc_anterior=p_proc_actual;
        nivel=fijar_nivel_int(NIVEL_3);

        /* El procés no segueix. Si hi hagués una replanificació pendent caldria
           desactivar-la ja que s'està realitzant en aquest precís moment */
        replanificacion_pendiente=0;
        eliminar_primero(&lista_listos);

        /* Si s ha especificat una llista desti per al BCP, s'inserta.
          Si lista_destino == &lista_listos -> c.contexto involuntari */
        if (lista_destino)
                insertar_ultimo(lista_destino, p_proc_anterior);

        p_proc_actual=planificador();

        /* S assigna una rodanxa completa al procés que s'executarà */
        p_proc_actual->rodaja=TICKS_POR_RODAJA;

        /* Si el procés ja ha acabat, no es salva i s'allibera la pila */
        if (p_proc_anterior->estado==TERMINADO) {
                contexto_aux=NULL;
                liberar_pila(p_proc_anterior->pila);
        }
        else
                contexto_aux=&(p_proc_anterior->contexto_regs);

        cambio_contexto(contexto_aux, &(p_proc_actual->contexto_regs));

        fijar_nivel_int(nivel);
}
```

- Alliberar un procés amb la funció anterior serà una feina molt més fàcil i només haurem de fer una crida a cambio_proceso amb valor NULL

```
/*
 *
 * Funció auxiliar que acaba el procés actual alliberant els seus recursos.
 * Feta servir per la crida termina_proceso i per rutines que tracten excepcions
 *
 */
static void liberar_proceso(){
        liberar_imagen(p_proc_actual->info_mem); /* alliberar mapa de memòria*/
        p_proc_actual->estado=TERMINADO;
        /* Sol·licita un canvi de procés però sense salvar l'estat del actual */
        cambio_proceso(NULL);
    return; /* En teoria no arribarà mai en aquest punt */
}
```

- Per últim només caldrà modificar petits aspectes per a que la pràctica anterior continuï funcionant.

```
/*
 * Tractament de la crida dormir
 */
int sis_dormir(){
        unsigned int segundos;

        segundos=(unsigned int)leer_registro(1);
        /* Establim quants TICKS de rellotge em d'esperar llegint el registre 1,
        ja que només se li ha passat una variable per paràmetre */
        p_proc_actual->plazo=segundos*TICK;
        /* En el procés actual se li actualitza el plaç d'espera en la taula BCP */
        /* es bloqueja el procés */
        p_proc_actual->estado=BLOQUEADO;
        cambio_proceso(&lista_dormidos);
        /* Cridem a la funció bloquejar per a "dormir" el procés actual */
        return 0; /* mai retornarà error */
}

//
```

ASSIGNATURA: SISTEMES OPERATIUS.

7.4.1 PLANIFICACIÓ PER PRIORITATS

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, afegint funcionalitats relacionades amb la planificació.
Aprofundir en la comprensió de la estructura interna del minikernel.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.
S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

L'objectiu de la següent pràctica és el d'incloure en el minikernel un algoritme de planificació basat en prioritats amb les següents característiques:

- Es tracta d'un sistema apropiatiu. No existeix el concepte de quàntum , i per tant, un procés en execució seguirà executant-se fins que acabi.
- La prioritat serà un valor enter entre 10 (mínima) i 50 (màxima)
- Cada procés tindrà una prioritat heretada del pare.
- El procés *init* té inicialment la prioritat mínima
- Un procés podrà canviar la seva prioritat amb la crida al sistema :

int fijar_prio(unsigned int prio( )

Retornarà -1 en cas d'error i 0 en cas contrari.

Repercussions sobre el nostre sistema simulat amb la inclusió del esquema de prioritats :

- El planificador es complica havent de buscar el procés amb major prioritat
- Apareixen nous tipus de canvis de context involuntaris. Sempre que es desperti un procés, ja sigui com a conseqüència d'una interrupció com de una crida al sistema amb major prioritat del procés actual, (mentre aquest roman en mode sistema) s'anotarà el canvi de context involuntari i s'aplicarà el mecanisme de les interrupcions *software.*

7.4.2 Comentaris i resolució

Continuant amb el tema de polítiques de planificació del processador, l'alumne haurà d'incorporar una planificació per prioritats, cada procés contindrà en el seu BCP la informació del seu grau de prioritat que com ja s'ha comentat els seus valors estaran entre 10 (els processos de menor nivell) i 50 (per els processos més prioritaris).

- Canvis en la estructura BCP indicant la prioritat del procés.

Modificacions sobre ***minikernel/kernel.c***

```
/*
 * Tractament de la crida per  fixar la prioritat
 */
int sis_fijar_prio(){
        unsigned int prioridad;
        // variable  entera per assignar la prioritat
        prioridad=(unsigned int)leer_registro(1);
        // la prioritat d'un procés serà assignada a través del primer paràmetre
        /* Si la prioritat es troba dins del rang vàlid */
        if ((prioridad<=MAX_PRIO) && (prioridad>=MIN_PRIO)) {
                /* guardem la prioritat al BCP */
                p_proc_actual->prioridad=prioridad;
                /* Comprovem si el procés ja no és el més prioritari */
                if (p_proc_actual !=maxima_prioridad(&lista_listos)) {
                        /* Si no ho és, canvi de context involuntari */
                        replanificacion_pendiente=1;
                        activar_int_SW();
                }
                return 0;
        }
        else
                return -1;
}
```

```
/*
 * Funció auxiliar que retorna el procés més prioritari d'una llista.
 */
static BCP * maxima_prioridad(lista_BCPs *lista) {
        BCP *proc_max;
```

```
        BCP *paux;

        /* El primer procés és el primer candidat */
        proc_max=lista->primero;
        paux=proc_max->siguiente;

        /* Recorrem la resta de la llista buscant "el procés de màxima prioritat" */
        while (paux!=NULL) {
                if (paux->prioridad > proc_max->prioridad)
                        proc_max=paux;
                paux=paux->siguiente;
        }
        return proc_max;
}
```

```
/*
 * Funció de planificació que implementa un algoritme basat en prioritats.
 */
static BCP * planificador(){
        while (lista_listos.primero==NULL)
                espera_int();           /* No s'ha de fer res */
        return maxima_prioridad(&lista_listos);
}
```

Modificar cambio_proceso ja que ara al no ser FIFO el procés no cal que estigui al principi de la llista

```
        replanificacion_pendiente=0;

        eliminar_elem(&lista_listos, p_proc_actual);

        /* Si se ha especificado una llista destino para el BCP, se inserta.
```

```
/*
 * Funció que desbloqueja el procés especificat
 */
static void desbloquear(BCP * proc, lista_BCPs *lista) {
        int nivel;

        proc->estado=LISTO;

        nivel=fijar_nivel_int(NIVEL_3);
        eliminar_elem(lista, proc);
        insertar_ultimo(&lista_listos, proc);
        fijar_nivel_int(nivel);
```

```
        /* Comprueva si el procés despertat és més prioritari */
        if (proc->prioridad > p_proc_actual->prioridad) {
                /* Si ho és, canvi de context involuntari */
                replanificacion_pendiente=1;
                activar_int_SW();
        }
}

/*
 * Funció que desbloqueja el procés méss prioritari d'una llista
 */
static void desbloquear_por_prioridad(lista_BCPs *lista) {
        BCP * proc;

        proc=maxima_prioridad(lista);
        desbloquear(proc, lista);
}
```

Treure actualizar_rodaja()

```
        p_proc->estado=LISTO;

                /* copia prio. de pare, menys init que no ho té (MIN_PRIO)*/
                if (p_proc_actual)
                        p_proc->prioridad=p_proc_actual->prioridad;
                else
                        p_proc->prioridad=MIN_PRIO;

                /* Dado que la rutina de int. de reloj también manipula la lista
```

ASSIGNATURA: SISTEMES OPERATIUS.

7.5.1 PLANIFICACIÓ TIPUS LINUX EN EL MINIKERNEL

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, afegint funcionalitats relacionades amb la planificació.
Aprofundir en la comprensió de la estructura interna del minikernel.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.
S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

L'objectiu de la següent pràctica es basarà en la introducció en el minikernel d'un sistema de planificació inspirat en un real, concretament el de Linux, per a que es pugui apreciar la complexitat dels algoritmes reals.

Es treballarà sobre la pràctica anterior modificant el sistema de prioritats.

Esquema del nostre nou sistema de prioritats :

- La prioritat efectiva d'un procés es calcula a partir de la seva prioritat base i el seu perfil d'execució.
- La prioritat efectiva inicial d'un procés és el mateixa que la seva prioritat base.
- Existeixen unitats d'apropiació (quàntum), però el seu tamany és proporcional a la prioritat efectiva del procés.
  (Exemple : prioritat efectiva = 40, quantum = 40 TICKS)
- En cada interrupció del rellotge es decrementa la prioritat efectiva del procés actual. Si arriba a 0, el procés en execució deixa el processador i es selecciona el procés amb major prioritat efectiva.

- Quan la prioritat efectiva de tots els processos llests per executar sigui igual a 0, es realitza un ajust de les prioritats de **tots els processos** mitjançant la següent formula:

  prioritat efectiva = prioritat efectiva / 2 + prioritat base

- Sempre que es desperti un procés s'ha de comparar la seva prioritat efectiva amb la actual i activar un canvi de context involuntari en cas que sigui major.

ASSIGNATURA: SISTEMES OPERATIUS.

7.3.1 IMPLEMENTACIÓ DE PROCESSOS LLEUGERS EN EL MINIKERNEL

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, desenvolupant noves crides al sistema.
Aprofundir en la comprensió de la estructura interna del minikernel.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.
S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

Comprendre els mecanismes de la creació dels processos lleugers
(o threads) serà la principal motivació de la següent pràctica.
Caldrà implementar un sistema per a crear un procés lleuger sobre el sistema la versió inicial del minikernel.

Per a la inclusió de processos lleugers en el minikernel només s'haurà d'incloure un nova crida al sistema per a crear-los:

int crear_thread (void *dir_funcio)

Aquesta rutina crea un flux d'execució que executa la retina que comença a l'adreça dir_funcio. Retornant 0 si no hi ha cap error i -1 en el cas contrari.
Per acabar l'execució d'un procés lleuger aquest utilitzarà la primitiva *terminar_proceso.*

Les noves directrius d'aquesta primitiva seran les següents :

- El minikernel portarà la compte de quants processos lleugers tenen cada procés.
- Quan el sistema operatiu rep la crida *terminar_proceso*, acaba la execució del procés lleuger actual i decrementa el contador de processos lleugers. Si aquest contador arriba a 0, el procés acaba completament.

Exemple de l'ús de primitives:

```
#include "servicios.h"
int funcio ( ){
...
}
int g ( ) {
...
terminar_proceso( );  // crida innecessaria
}
int main( ){
....
crear_thread(funcio);
....
crear thread(g);
....
}
```

ESPECIFICACIONS

Per a la facilitació de la resolució, caldrà remarcar canvis a tenir en compte a la hora de la inclusió de processos lleugers:

- S'ha de dissenyar les estructures de dades adequades per crear aquesta nova abstracció :

- Definir, a més del vector de BCP, un vector de processos lleugers, amb un tamany màxim de MAX_THREADS que emmagatzemarà la informació de tots els processos lleugers existents en el sistema

- Si es considera necessari, els processos lleugers d'un mateix procés podrien estar enllaçats en una llista apuntada des de la BCP.
- El BCT ha d'incloure una referència al BCP corresponent per poder accedir a la informació global del procés.
- Serà necessari determinar quins aspectes estan vinculats al procés (BCP) i quins estan relacionats amb un procés lleuger (BCT)

- Hi hauran canvis sobre el planificador ja que aquest passarà de planificar sobre processos normals a fer-ho sobre processos lleugers. Per tant totes les llistes de planificació (llests, bloquejats, etc) seran BCT en comptes de BCP.
- És important que es remarqui la diferencia entre la creació del procés pare i un procés fill, sent *crear_proceso* (procés pare) i *crear_thread* (procés fill) les seves crides al sistema.

ASSIGNATURA: SISTEMES OPERATIUS.

7.4.1 DISSENY DE SEMAFORS EN EL MINIKERNEL

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, del sistema simulat minikernel, desenvolupant noves crides al sistema.
Aprofundir en la comprensió de la estructura interna del minikernel.

MATERIAL

Per a la realització de la pràctica es treballarà amb Linux executant el codi de suport del minikernel entregat.
S'hauran de modificar els arxius situats als directoris que cregueu oportuns:

- BOOT (sistema d'arranc)
- MINIKERNEL (cos del sistema)
- USUARI (programes d'usuari)

DESCRIPCIÓ

Per a completar el nostre sistema operatiu simulat s'afegirà un mecanisme de mutex (semàfors).

S'hauran d'implementar dos tipus de *mutex:*

1- *Mutex* no recursius
Si un procés que posseeix un *mutex,* inteta ser bloquejar un altre cop, aquest retornarà un error, ja que s'està produint un cas trivial d'interbloqueig.

2- *Mutex* recursius
Podrà permetre ser bloquejat tantes vegades com es vulgui. El *mutex* només quedarà desbloquejat si s'allibera com tantes vegades hagi estat bloquejat.

Recordem que un **mutex** és un mecanisme de sincronització per a l'accés exclusiu a recursos compartits i per assegurar l'exclusió mútua sobre seccions critiques.

Les dues operacions bàsiques les definirem com a :

- **Lock :** bloquejar el *mutex*.. Si el mutex ja està bloquejat per un altre procés, el procés que realitza l'operació es bloqueja. En cas contrari, es bloqueja el *mutex* sense bloquejar el procés.
- **Unlock :** desbloquejar el *mutex*. Si existeixen processos bloquejats en ell, es desbloquejarà un d'ells, que serà el nou procés que adquireixi el *mutex*

Pseudocodi que utilitza un ***mutex*** per a protegir l'accés a una secció critica

Lock(m);

< secció critica >

Unlock(m);

L'interface dels serveis de mutex seran els següents:

int crear_mutex(char *nom, int tipus);

Crearà el *mutex* amb el nom i el tipus donats, retornant un enter que representarà un descriptor per accedir al *mutex.* En cas d'error, retornarà un valor negatiu.

Per facilitar la especificació del tipus el primer pas de la pràctica consistirà en la inclusió de *mutex* dins dels fitxers de capçalera dels programes d'usuari (servicios.h) i el del sistema operatiu (kernel.h).

#define NO_RECURSIU 0

#define RECURSIU 1

Nota : el mutex no s'obre al crear-lo, sinó que per obrir-lo s'haurà de cridar explícitament a la funció (obrir_mutex).

8. SIMULADORS

8.1 Simulador Planificació de processos

La aportació del treball "**Análisis de diferentes tipos de planificación para sistema a tiempo real y no real"** per els alumnes Joan Gemio Valero1, Jordi Garcia Rincon1, Francesc Parramon Muntada1, Isaac Sanmarti Sabates de Telecomunicació ha permès la introducció d'un sistema de simulació per a poder fer proves i comprendre els diferents esdeveniments que es produeixen en el planificador.

El sistema ha estat desenvolupat en MATLAB i permet simular diferent politiques de planificació creant els postres propis processos indicant el seu temps d'arribada al sistema i indicar quan aquests efectuen entrada/sortida.

El simulador contempla les següents politiques de planificació:

- **FIFO (First Input First Output)**
- **SPN (Shortest Process Next)**
- **HRRN (Highest Response Ratio Next)**
- **SRT (Shortest Remaining Time)**
- **RR (Round Robin)**
- **Prioridades**

Donat que la versió inicial no contempla tots els cassos que es poden produir específicament en la política **SPN (Shortest Process Next)**, s'han afegit dos modificacions per a contemplar el temps d'arribada dels processos per tal d'oferir a la cpu el procés que ha arribat en primer lloc o en una altre versió el que ha arribat en últim lloc, d'aquesta manera l'alumne podrà experimentar i contemplar el diferent comportament que es produeixen al canviar propietats bàsiques del algoritme de planificació.

Exemple del funcionament del Simulador de processos :

Creació dels processos

Tasca1=[1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1];

Tasca 2=[1 1 1 1 1 1 1 1 1 1 1];

Tasca 3=[1 1 1 2 2 2 1 1 1 2 2 2 1 1 1];

Tasca 4=[1 1 1 2 2 2 1 1 1 2 2 2 1 1 1];

1 - Procés en execució

2 - Entrada/sortida

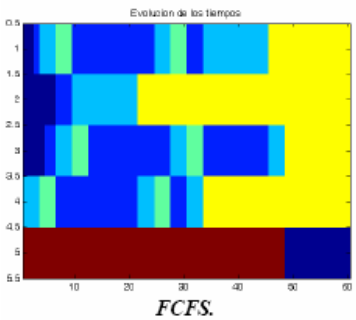


Figura Execució del planificador fent crida a la política FCFS

9. PRÀCTIQUES CLÀSSIQUES

ASSIGNATURA: SISTEMES OPERATIUS.

9.1 Shell Scripts

CURS: 2006/2007

OBJETIU

Habituar-se a l'ús, així com conèixer les potencialitats, de sistemes operatius tipus UNIX a través de la programació d'arxius de comandes (shell scripts).

MATERIAL

Per a la realització de la pràctica es treballarà amb l'interpret de comandes (shell) cshell.
L'ajuda a utilitzar la trobareu a:

- La comanda man de unix: man [-s <secció>] <comando>
  Exem: man grep o man awk.
- Trobareu ajuda i referències sobre la programació en shell script a la següent url: http://mmm.uab.es/soremo/apuntes.html

DESCRIPCIÓ – PART 1

En aquest primer exercici de shell script ha de ser capaç d'analitzar les estructures sintàctiques d'un conjunt de dades emmagatzemades dins un arxiu (procés conegut com parsing).

FUNCIONAMENT.

Disposarem de fitxers, els quals contindran informació relacionada amb jugadors de futbol d'un determinat club i el seu pressupost, seguint aquest arxiu una estructura com la que s'indicarà a continuació:

Fitxer Club

- Primera línia : <pressupost_del_club >

on: pressupost_del_club serà un numero sencer que indicarà el pressupost que el club destinarà a la compra de jugadors

Les següents línies tindran la següent estructura sintàctica.

- <Nom_Cognom> : <nacionalitat> : <posició_de_joc> : <Preu_de_venta> : <situació_al_camp>

On: *Nom_Cognom* (indicarà el nom o sobrenom del jugador), *nacionalitat* (abreviació de la nacionalitat amb 3 caràcters), *posició_de_joc* (poden prendre 4 valors predefinits : Porter, Defensa, Centrecampista i Davanter), *preu_de_venta* (preu actual del jugador en el mercat) i *situació_en_el_camp* (podent prendre dos valors "En_joc" o "suplent")

Exemple de fitxer club :

```
10040000
Albert_Jorquera:ESP:Porter:20000:En_Joc
Victor_Valdes:ESP:Porter:20000:suplent
Ruben_Martinez:ESP:Porter:20000:suplent
Juliano_Belletti:BRA:Defensa:30000:suplent
Edmilson:BRA:Defensa:30000:suplent
Rafael_Marquez:MEX:Defensa:30000:En_Joc
Oleguer_Presas_Renom:ESP:Defensa:30000:En_Joc
Carles_Puyol:ESP:Defensa:30000:En_Joc
Sylvinho:BRA:Defensa:30000:suplent
Lilian_Thuram:FRA:Defensa:30000:En_Joc
Giovanni_Van_Bronckhorst:HOL:Defensa:30000:suplent
Gianluca_Zambrotta:ITA:Defensa:30000:En_Joc
Deco_Sousa:POR:Centrecampista:40000:En_Joc
Giovanni_Dos_Santos:MEX:Centrecampista:40000:suplent
Ludovic_Giuly:FRA:Centrecampista:40000:suplent
Andres_Iniesta:ESP:Centrecampista:40000:suplent
Thiago_Motta:BRA:Centrecampista:40000:suplent
Ronaldinho_Gaucho:BRA:Centrecampista:40000:En_Joc
Xavi_Creus:ESP:Centrecampista:40000:En_Joc
Samuel_Etoo:CAM:Davanter:50000:En_Joc
Marin_Santiago_Ezquerro:ESP:Tallantar:50000:suplent
Eidur_Gudjohnsen:ISL:Tallantar:50000:suplent
Lionel_Messi:ARG:Tallantar:50000:En_Joc
Javier_Saviola:ARG:Tallantar:50000:suplent
```

Finalment el shell script que caldrà desenvolupar ha de llegir el fitxer club, que podria contenir una estructura sintàctica errònia deguda a una mala construcció d'un o més jugadors, per a produir un nou arxiu club sense errors, durant l'anàlisi haurà de mostrar la línia errònia i indicar quanta camps falten en la seva construcció.

Així doncs, un cop analitzat per el shell script de verificació hauria de donar les següents sortides per pantalla:

Exemple de jugadors amb construcció errònia i sortides per pantalla del shell :

```
Samuel_Etoo:Davanter:50000:En_Joc            (falta un paràmetre)
Marin_Santiago_Ezquerro:ESP:Davanter:50000:suplent:Lesionat   (sobra 1 paràmetre)
Eidur_Gudjohnsen    (jugador mal construït)
…
```

Un cop s'ha corregit el fitxer erroni i el shell ha verificat que aquest no conté cap jugador amb una estructura sintàctica errònia, analitzarem la correcta construcció del equip.

Per a poder fer aquesta comprovació caldrà verificar :

- L'equip conté 11 jugadors en joc (camp *situació_en_el_camp* En_joc)
- L'equip posseeix almenys un porter en joc

En el cas que alguna de les anteriors comprovacions doni un resultat erroni caldrà mostrar-lo per pantalla.

Sent les possibles sortides:

1- Error L'equip ha de tenir 11 jugadors en joc

2- L'equip ha de tenir com a mínim un porter

3- Equip incorrecte falten jugadors. Un equip de futbol ha de tenir com a mínim 11 jugadors

4- Equip correctament format

DESCRIPCIÓ – PART 2

En aquest segon exercici haureu de desenvolupar un pseudo-mercat virtual de jugadors, que permetran als clubs vendre i comprar jugadors disponibles segons el seu pressupost.

FUNCIONAMENT.

Donada l'estructura sintàctica del exercici anterior, haureu de realitzar les següents funcions:

- Canvi de la situació del joc
- Vendre jugadors (Exemple crida: Vendre Nom Fitxer_mercat)
- Comprar jugadors (Exemple crida: Vendre Nom Fitxer_mercat)
- Canvi <Nom jugador> <club>

L'script proposat serà l'encarregat d'efectuar els canvis en el camp *situació_en_el_camp*, sent aquest el que farà els canvis entre $En\_joc \longleftrightarrow Suplent$

Basant-se en la idea del pressupost descriurem el funcionament de cada una de les funcions que es demanen implementar:

- vendre <Nom Jugador> <Fitxer Mercat>

Aquest script reestructurarà el fitxer club eliminant el jugador venut i incrementarà la quantitat del pressupost amb el valor d'aquest jugador.
Nota : Haureu de detectar en quin equip es troba per a realitzar els canvis oportuns.

- Comprar <Nom> <club> <Fitxer Mercat>

Aquest script sostraurà del fitxer mercat el jugador proposat a ser comprat. El fitxer club serà reestructurat incloent aquest jugador i el pressupost del club serà reduït amb la quantitat del preu del jugador.

Un cop venut o comprat el shell haurà de cridar a la funció de verificació de plantilla per si es necessiten efectuar canvis en el camp *situació_en_el_camp*.

10.CONCLUSIONS

Descripció del desenvolupament del projecte

- S'ha dissenyat i desenvolupat un entorn de gestió documental anomenat Sistema Gestor de Pràctiques per al emmagatzemament d'informació docent.
- S'ha instal·lat l'entorn sobre un sistema Linux i Windows, per assolir els requeriments funcionals i de dades, s'han dut a terme diverses implementacions i modificacions de l'entorn, que es poden resumir en les següents :
    - Modificació de la gestió d'usuaris, creant i modificant rols d'usuaris, i afegint noves característiques a la gestió de fitxers.
- S'han portat a terme diverses proves sobre les funcionalitats creades, en diferents assignatures.
- S'han analitzat i adaptat diferents pràctiques sota un nou entorn de treball per a aprofundir en temes no tractats fins el moment en les classes practiques.
- S'ha modificat un simulador realitzat per un alumne de Telecomunicacions per a que realitzés les funcions especificades de manera correcte per tal d'adaptar-lo i ampliar el nou conjunt de temes tractats en les classes de pràctiques.

Conclusions del treball realitzat

- Amb la realització del projecte he aprofundit els coneixements en l'administració de servidors Linux i Windows, en especial en l'administració d'entorns LAMP (Linux Apache MySql Php) alhora que he ampliat la meva experiència en la programació en els llenguatges PHP, HTML i javascript.
- Les noves pràctiques han estat adaptades per a una bona comprensió permetent una rapida realització i donant la possibilitat de tractar més temes durant les sessions.
- La via d'aprenentatge del temari de les sessions teòriques, és poden assimilar millor amb les realitzacions de simuladors.
- Aquest projecte m'ha ajudat a entendre el grau de responsabilitat que els professors tenen i la dificil tasca en que es troben al volguer incorporar nou temari.

Treballs futurs.

- El nou entorn, degut al seu disseny i a les diferents comunitats d'usuaris a la que pot anar dirigit, ofereix unes possibilitats molt altes d'adaptació a les diferents necessitats que poden sorgir dins d'un ambient docent.

11.BIBLIOGRAFIA I REFERÈNCIES

[UAB] http://uab.es

[ARCOS] http://www.arcos.inf.uc3m.es/~ssoo-va/

[PHP] http://www.php.net

[CGP02] J. Carretero, F. García, F. Pérez, **Prácticas de Sistemas Operativos,** Gener 2002

RESUM:

En aquest projecte s'ha desenvolupat un entorn repositori de pràctiques per a emmagatzemar i donar una eina de control del material docent que es disposa. El desenvolupament dels mòduls del sistema s'han realitzat després de contemplar els diferents rols que existeixen dintre dels membres del personal docent, sent aquest becaris o professors. Donat que el personal destinat a l'ensenyament està sotmès a un alt grau de rotacions aquest sistema intentarà ser un punt de trobada de totes les seves aportacions, ajudant així la tasca d'avaluació i explicació de les pràctiques. L'estudi d'un nou ventall de pràctiques ha permès conèixer un nou entorn de treball, que ofereix camps no tractats fins al moment a través de la implantació d'un sistema operatiu simulat per sobre del sistema operatiu Linux.

RESUMEN:

En este proyecto se ha desarrollado un entorno repositorio de prácticas para almacenar y dar una herramienta de control del material docente que se dispone. El desarrollo de los módulos del sistema se han realizado tras contemplar los diferentes roles que existen dentro de los miembros del personal docente, siendo estos becarios o profesores. Dado que el personal destinado a la enseñanza está sometido a un alto grado de rotaciones este sistema intentará ser un punto de encuentro de todas sus aportaciones, ayudando así la tarea de evaluación y explicación de las prácticas. El estudio de un nuevo repertorio de prácticas ha permitido conocer un nuevo entorno de trabajo, que ofrece campos no tratados hasta el momento a través de la implantación de un sistema operativo simulado por encima del sistema operativo Linux.

ABSTRACT:

In this project a repository's system of practical exercises has been developed to store and give a management tool for the available educational material as well as the one to be developed in the future. The development of the modules of this system has been made after analyzing the different roles that exist within the members of the faculty, being these scholarship holders or professors. Given that laboratory professors show a high degree of rotation, this system will centralize all these contributions, helping to the task of evaluation and explanation of the practical exercises. The design and development of a new set of practical exercises has allowed the author to know a new workspace that offers issues non-treated until this moment through the implantation of a simulated operating system running on Linux.