



Universidad
Nacional
de Córdoba

Cátedra de Sistemas Operativos II

Trabajo Práctico N° I

Bosack Federico Ezequiel
23 de abril del 2020



Índice

Introducción	3
Propósito	3
Ámbito del sistema	3
Definiciones, Acrónimos y Abreviaturas	4
Referencias	4
Descripción general del documento	4
Descripción General	4
Perspectiva del producto	4
Funciones del producto	5
Características de los usuarios	5
Restricciones	5
Suposiciones y Dependencias	5
Requisitos Futuros	5
Requisitos Específicos	6
Interfaces Externas	6
Funciones	6
Requisitos de Rendimiento	7
Restricciones de Diseño	7
Atributos del Sistema	7
Otros Requisitos	7
Diseño de solución	8
Implementación y Resultados	9
Conclusiones	12
Apéndices	13

Introducción

Propósito

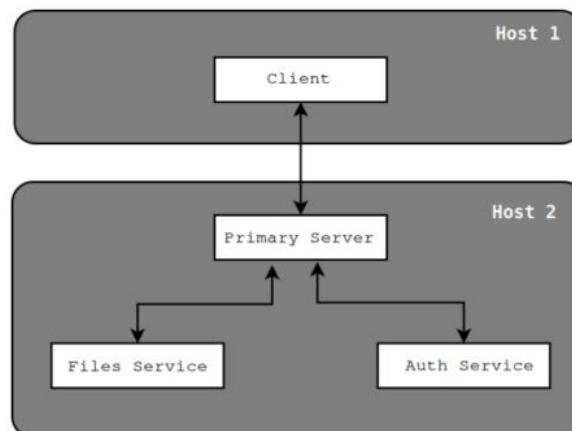
El propósito del trabajo es desarrollar un software que haga uso de las API de IPC del Sistema Operativo, implementando lo visto en el teórico, práctico y haciendo uso todos los conocimientos adquiridos en Ingeniería de Software y Sistemas Operativos I.

El propósito del sistema es que permita realizar la conexión, control y transferencia de datos entre un proceso cliente y otro proceso servidor, ubicados en dos Host distintos.

Ámbito del sistema

El sistema correrá en un entorno de Linux. Se podrá ejecutar a través de la interfaz de loopback, o el cliente y el servidor ubicados en dos hosts distintos, por ejemplo el servidor en una computadora y el cliente en una placa RaspberryPI.

El cliente se conectará al servidor una vez habiendo validado su usuario y contraseña únicamente, y el servidor podrá atender un solo cliente.



Definiciones, Acrónimos y Abreviaturas

- **Primary_server:** Servidor primario encargado de la comunicación con el cliente a través de un socket, y de proveer las funcionalidades comunicándose con los procesos auth_service y file_service.
- **Auth_service:** Proceso encargado de proveer toda la funcionalidad asociada al manejo de la base de datos de usuarios, validar, bloquear, listar y cambiar contraseña
- **File_service:** Proceso encargado de proveer las funcionalidades de listar las imágenes disponibles para la descarga, y de crear un nuevo socket para enviar la imagen a descargar al cliente.

Referencias

Se indican a continuación dos referencias propuestas en clase.

1. <https://gcc.gnu.org/onlinedocs/gcc-9.3.0/gcc.pdf>
2. <https://gcc.gnu.org/onlinedocs/gcc-9.3.0/gcc/Common-Type-Attributes.html#Common-Type-Attributes>

Descripción general del documento

El documento está dividido en dos carpetas, cliente y servidor, ambas carpetas están separadas y se ejecutan por separado.

Tanto el cliente como el servidor tienen su propio Makefile , Funciones y librerías.

Descripción General

Perspectiva del producto

El producto está orientado a la comunicación entre un cliente y un servidor, y este último con dos procesos.

La comunicación entre el Cliente (Proceso 1) y el Primary Server (Proceso 2) se implementa por medio de sockets, en este trabajo se implementó el socket INET con conexión segura (TCP)-



Funciones del producto

Las funciones del producto se dividirá por las funciones de cada proceso:

Cliente: Su función principal es establecer la conexión con el Primary Server, y ejecutar comandos que serán atendidos por éste, también crear un socket recibir la imagen y grabarla en un pendrive.

Servidor:

Características de los usuarios

El usuario que ejecutará el cliente debe estar registrado en la base de datos del servidor y no estar bloqueado, y tambien debera tener un conocimiento mínimo del las funcionalidades del software como las limitaciones.

Restricciones

- No se garantiza la encriptación de los datos enviados.
- La contraseña será visible en el prompt.
- Una vez bloqueado un usuario, solo accediendo a la base de datos del servidor será posible desbloquearlo.

Suposiciones y Dependencias

Los programas servidor y cliente correrán en sistemas operativos con kernel Linux.

Tanto el servidor como el cliente contará con una dirección ipv4 y con acceso a internet a través de un router o modem.

El cliente tendrá conectado con un pendrive para poder grabar la imagen.

Requisitos Futuros

- Encriptación de las contraseñas.
- Update: Enviar un binario para actualización del software.
- Autenticación local en el cliente.

Requisitos Específicos

Interfaces Externas

- **Ciente** En caso de ser una Raspberry PI debe tener instalado un sistema operativo con kernel linux y se tendrá que copiar la carpeta del cliente y luego ejecutarla, puede ser con ssh.
- **Servidor:** También debe tener instalado un sistema operativo con kernel linux. y se ejecutará desde la carpeta servidor.

Funciones


- **Ciente:** Su función principal es establecer la conexión con el Primary Server, y ejecutar comandos que serán atendidos por esté.

Los comandos que puede enviar y serán reconocidos son:

1. **exit:** cerrar conexión con el Primary Server.
2. **user ls:** Le indica al Primary Server que liste los usuarios y sus detalles. El cliente los tomara como mensajes del servidor por lo que no contara con una función especial para recibir la lista.
3. **user passwd <new pass>:** Le indica al Primary Server que cambiara la contraseña del usuario actual por new pass
4. **file ls:** Le indica al Primary Server que liste las imágenes disponibles a descargar y su hash MD5.
5. **file down <image name>:** Le indica al Primary Server que comience la transferencia de la imagen.

El cliente estará encargado crear un socket servidor, para que el socket cliente creado por el File Service se conecte y envíe la imagen a la vez que la graba un pendrive y luego calcular su hash MD5 y mostrarlo con la tabla de particiones.

- **Primary server:** Es el Servicio encargado de establecer conexiones y recibir comandos del cliente. Hace uso de los otros 2 procesos para cumplir los requerimientos del software.



Esté a través del mecanismo de IPC de Colas de mensajes (SysV) se comunicara con los otros dos procesos, explicados a continuación.

- **Auth Service:** Encargado de proveer toda la funcionalidad asociada al manejo de la base de datos de usuarios:
 - validar usuarios, bloquear usuario, listar usuarios y cambiar la contraseña de los mismos.
- **File Service:** Encargado de proveer funcionalidad relacionada a las imágenes disponibles para descarga, cómo obtener nombres, tamaño y sus hash MD5.

También es el encargado de creando un socket cliente establecer la conexión con el socket servidor creado por el cliente y transferir la imagen.

Requisitos de Rendimiento

- No especificados.

Restricciones de Diseño

- Utilizar las API de IPC.
- Desarrollo en C.
- Sistema de autenticación de usuario y contraseña.
- Implementar sockets distintos para la transferencia de la imagen.
- Los procesos deben ser 3 binarios distintos en el caso del servidor.
- El servidor solo puede atender un cliente a la vez.

Atributos del Sistema

El sistema cuenta con un servidor, un cliente y conexión a internet.

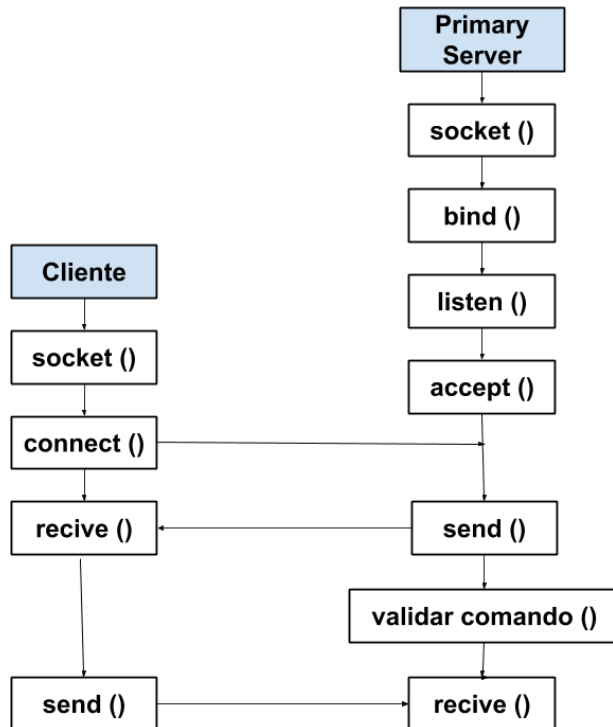
Otros Requisitos

- Los procesos deben ser mono-thread.
- Sistema operativo con kernel linux.
- Fiabilidad de los datos.
- Control de errores en la transferencia de archivos.

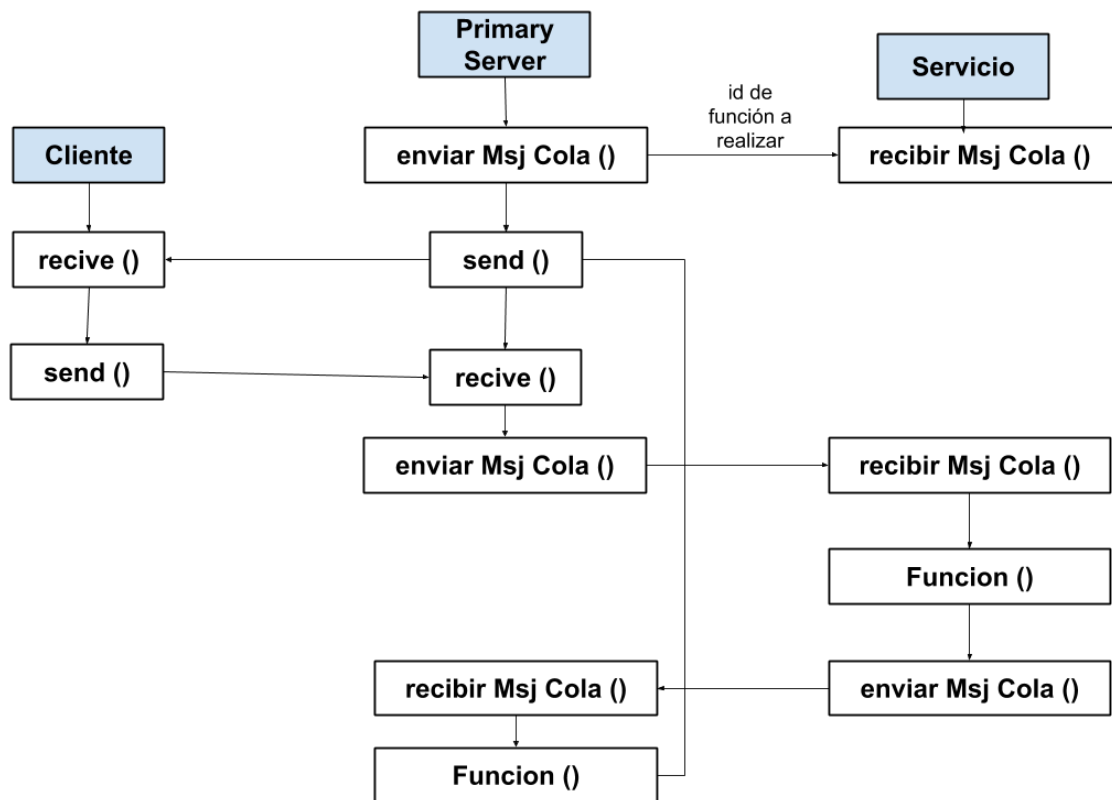
Diseño de solución

Se realizó la solución comenzando la comunicación entre el cliente y el Primary server a través de socket INET, se comenzó testeando en la interfaz de loopback en dos terminales distintas.

Protocolo de comunicación cliente-servidor



Protocolo de comunicación cliente-servidor-servicio(Auth o File)



Implementación y Resultados

A continuación se muestran los resultados de la implementación de las diferentes funcionalidades.

- Base de datos en formato .csv

```
usuarios.csv (~/Escritorio/Facultad/Facultad2020/SOII/practic...
Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda

fedebos,1234,true,H:20:1_D:23/4/2020
fedebos1,1234,true,H:12:19_D:17/4/2020
fedebos2,1234,true,H:12:26_D:17/4/2020
fedebos3,12345,false,H:19:10_D:23/4/2020

CSV  Ancho de la tabulación: 4  Ln 4, Col 41  INS
```

- **Autenticación del cliente**

<pre>fedebos@fedebos:~/Escritorio/Facultad/Facultad2020/S0II/pract ico/TPS2020/TP1/cliente\$./bin/client 127.0.0.1 8080 Usuario,Contraseña: fedebos,1234 Comando:█</pre>	<pre>fedebos@fedebos:~/Escritorio/Facultad/Facultad2020/S0II/pract ico/TPS2020/TP1/servidor\$./bin/server 8080 Proceso: 1680 - socket disponible: 8080 SERVIDOR: Cliente conectado █</pre>
---	---

- **user ls:** debido al protocolo de comunicación, para listar los usuario, el cliente tiene que mandar un mensaje al primary server por cada usuario listado (con enter alcanza)

<pre>fedebos@fedebos:~/Escritorio/Facultad/Facultad2020/S0II/pract ico/TPS2020/TP1/cliente\$./bin/client 127.0.0.1 8080 Usuario,Contraseña: fedebos,1234 Comando:user ls Usuarios Activo Ultima vez fedebos true H:20:11 D:23/4/2020 fedebos1 true H:12:19 D:17/4/2020 fedebos2 true H:12:26 D:17/4/2020 fedebos3 false H:19:10 D:23/4/2020 Comando:█</pre>	<pre>fedebos@fedebos:~/Escritorio/Facultad/Facultad2020/S0II/pract ico/TPS2020/TP1/servidor\$./bin/server 8080 Proceso: 1680 - socket disponible: 8080 SERVIDOR: Cliente conectado PROCESO 1680. Recibí: user ls █</pre>
--	---

- **user passwd <nueva_contraseña>**

<pre>Comando:user passwd Contra55 Contraseña Cambiada Comando:█</pre>	<pre>SERVIDOR: Cliente conectado PROCESO 2140. Recibí: user passwd Contra55 █</pre>
---	---

Nueva base de datos

```
fedebos,Contra55,true,H:20:8_D:23/4/2020
fedebos1,1234,true,H:12:19_D:17/4/2020
fedebos2,1234,true,H:12:26_D:17/4/2020
fedebos3,12345,false,H:19:10_D:23/4/2020
```

- **File ls:** Debido al protocolo de comunicación, también en esta función el tiene que mandar un mensaje al primary server por cada imagen y hash MD5 listado (con enter alcanza)

<pre> Comando:file ls Listado : Nombre - Tamaño tahr-6.0.5 PAE.iso - 205 mb (enter y esperar MD5) MD5: 902e1b3bb6999a6269271a0e54305efe (enter) xenialpup-7.5-uefi.iso - 324 mb (enter y esperar MD5) MD5: cd40febf96a102c9ade6b8e96a3a95c3 (enter) bionicpup32-8.0-uefi.iso - 273 mb (enter y esperar MD5) MD5: dbf1b6dd0e9a5bdfdc7662eb0820a46d (enter) Comando: </pre>	<pre> PROCESO 2140. Recibi: file ls </pre>
---	--

- **File down:** Al usar el comando el servidor crea un socket **cliente** que se conectara por el puerto preestablecido 8080 con el **servidor** que creo el cliente, esté recibirá y lo escribirá en el pendrive, luego mostrara su hash y su tabla de particiones.

<pre> fedebos@fedebos:~/Escritorio/Facultad/Facultad2020/S0II/practico/TPS2020/TP1_FBosack_392675650/cliente\$./bin/client 127.0.0.1 2020 Usuario,Contraseña: fedebos,1234 Comando:file ls Listado : Nombre - Tamaño tahr-6.0.5 PAE.iso - 205 mb (enter y esperar MD5) MD5: 902e1b3bb6999a6269271a0e54305efe (enter) xenialpup-7.5-uefi.iso - 324 mb (enter y esperar MD5) MD5: cd40febf96a102c9ade6b8e96a3a95c3 (enter) bionicpup32-8.0-uefi.iso - 273 mb (enter y esperar MD5) MD5: dbf1b6dd0e9a5bdfdc7662eb0820a46d (enter) Comando:file down tahr-6.0.5_PAE.iso Comenzar(enter) Socket disponible: 8080 MD5: 902e1b3bb6999a6269271a0e54305efe Comando: </pre>	<pre> fedebos@fedebos:~/Escritorio/Facultad/Facultad2020/S0II/practico/TPS2020/TP1_FBosack_392675650/servidor\$./bin/primary_server 2020 Proceso: 5874 - socket disponible: 2020 SERVIDOR: Cliente conectado PROCESO 5874. Recibi: file ls PROCESO 5874. Recibi: file down tahr-6.0.5_PAE.iso </pre>
--	---



Conclusiones

Mediante la realización del trabajo práctico se pudo consolidar lo aprendido en las clases teóricas y prácticas sobre socket y como aplicar comunicaciones entre procesos (IPC). Se pudieron llevar a la práctica los conceptos adquiridos logrando cumplir con el objetivo planteado en la consigna.

Fue difícil limitarse a 4 procesos (1 cliente,3 server), pero se pudo lograr, al momento de la escritura en el pendrive fue donde más inconvenientes se presentaron. Creo que con más tiempo el código y el software se puede ir actualizando y mejorando.

Hay funcionalidades que quizas la forma de resolverlo no fue la mas optima, pero cumple con su función y es donde creo que trabajo a trabajo voy a ir mejorando en ese aspecto.



Apéndices

- El material brindado en clase por los docentes, como las consultas y links prestados por slack fueron de gran ayuda.
- [How to calculate the MD5 hash of a large file in C?](#)
- [Colas de mensajes en C para Linux](#)

