

Curso Linux

UNIDAD I

[Unidad I@presentación]#

1.1 Instalación de Linux como cliente y servidor

1.2 Instalación de software

1.3 Interfaces gráficas KDE y GNOME

1.4 Aplicaciones de uso común

1.5 Lenguajes de programación

[Instalación@Unidad I]\$

- Realizar respaldo de archivos y configuraciones.
- Si se desea tener un sistema de arranque múltiple instalar primero los sistemas operativos y hasta el final Linux.
- Realizar particionado de preferencia manual antes de la instalación.

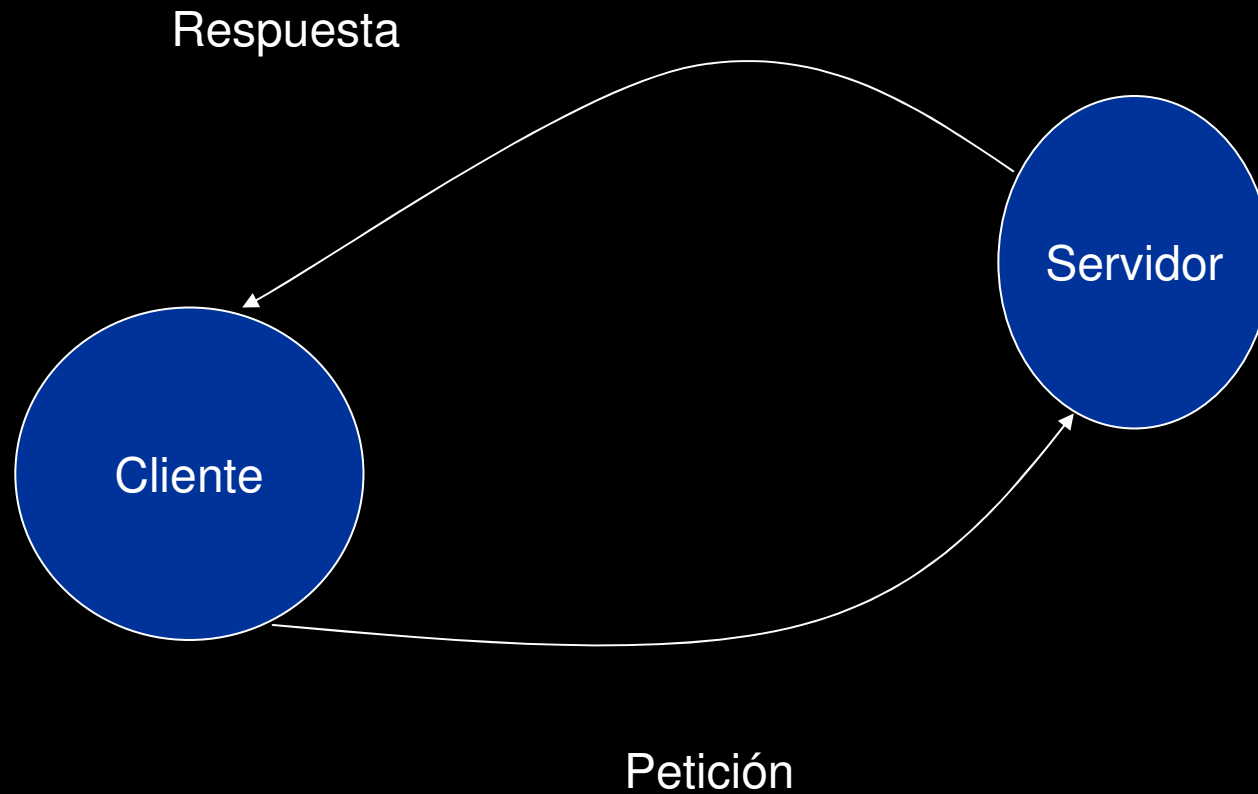
[Instalación@Unidad I]\$

- Se deberá elegir el tipo de instalación: modo gráfico, modo texto o modo experto.
- Se deberá tener conocimiento sobre el hardware del equipo.
- Se deberá saber que uso se le dará a la máquina para así seleccionar el tipo de instalación y conocer los paquetes a instalar.

[Instalación@Unidad I]\$

- La mayoría de las distribuciones de Linux cuentan con instaladores para:
 - Clientes
 - Estaciones de Trabajo
 - Servidores
- Se recomienda en servidor no instalar interfaz gráfica y paquetes que no se usan
- El problema no es instalar Linux sino configurarlo

[Cliente/Servidor@Unidad I]\$



[Computadoras@Unidad I]\$

- El término cliente/servidor se refiere a procesos pero se asocia con máquinas
- Supercomputadoras
- Mainframes (macrocomputadoras)
- Estaciones de trabajos (minicomputadoras)
- Computadoras personales (microcomputadoras).

[Cliente/Servidor@Unidad I]\$

- Servidor: a computadoras de gran tamaño, brindan servicios a otras máquinas.
- Cliente: computadoras de menor tamaño, limitadas en recursos, reciben servicios de los servidores. Terminales tontas.
- Estaciones de trabajo: computadoras de mayores prestaciones que los clientes pero menores que los servidores.

[Actualización@Unidad I]\$

- La actualización es más lenta. Es mejor realizar un respaldo y luego instalar.
- Particiones: /, /usr, /var, /home, /tmp, /boot y Swap.
- Disk druid es un asistente para particionado.
- NTFS es más estricto a la hora de realizar modificaciones en las particiones.

[Unidad I@presentación]#

1.1 Instalación de Linux como cliente y servidor

1.2 Instalación de software

1.3 Interfaces gráficas KDE y GNOME

1.4 Aplicaciones de uso común

1.5 Lenguajes de programación

[Instalación de paquetes@Unidad I]\$

- `tar -xzvf paquete.tar.gz`
- `cd paquete`
- `./configure`
- `make`
- `make install`

[Instalación de sw@Unidad I]\$

- rpm -i paquete
- rpm -U actualiza paquete

- rpm -qa
- rpm -qi
- rpm -qf

- rpm -e desinstala paquete

[Configuración SW@Unidad I]\$

- Linux no cuenta con un registro pero si con archivos de configuración. Las primeras versiones de Windows no tenían Registro (archivos .INI)
- La configuración depende de cada servicio.
- La localización de los archivos de configuración puede variar en cada distribución

[apt@Unidad I]\$

- Advanced Packaging Tool.
- /etc/apt/sources.list
- apt-get install paquete
- apt-get remove paquete
- apt-get --reinstall install paquete
- apt-cdrom ruta add
- apt-get -u install lilo

[Unidad I@presentación]#

1.1 Instalación de Linux como cliente y servidor

1.2 Instalación de software

1.3 Interfaces gráficas KDE y GNOME

1.4 Aplicaciones de uso común

1.5 Lenguajes de programación

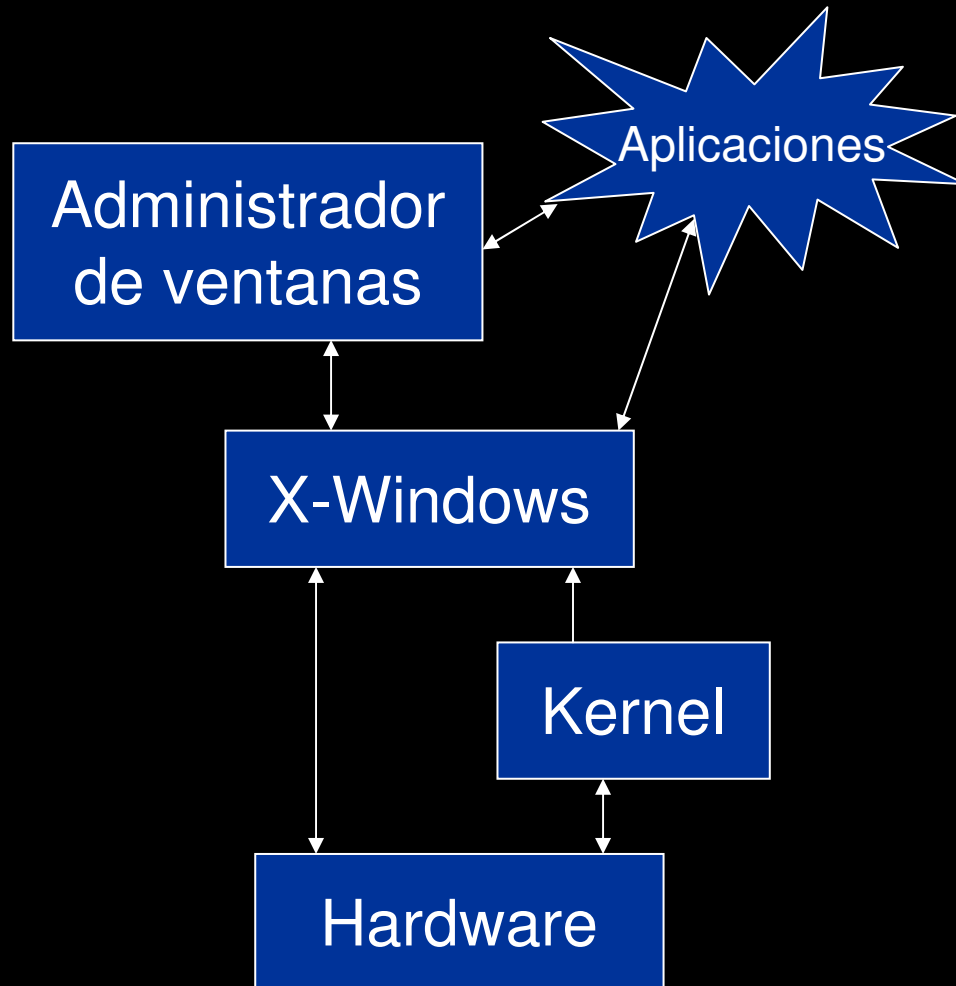
[GUI@Unidad I]\$

- Linux soporta dos tipos de interfaces de usuario: modo texto y modo gráfico.
- La interfaz en modo texto llamada Shell es poco intuitiva para el usuario final.
- La interfaz gráfica es más intuitiva pero requiere de mayor capacidad en el hardware.

[GUI@Unidad I]\$

- La GUI se basan en el X-Windows, el cual está basado en el paradigma cliente/servidor y es independiente del nucleo.
- X-Windows sólo define las directivas gráficas (puntos, líneas, cuadros, etc.) y no administra ventanas. El manejador de ventanas está aparte.

[GUI@Unidad I]\$



[GUI@Unidad I]\$

- XFree86
- Enlightenment
- WindowMaker
- Blackbox
- FluxBox
- GNOME
- KDE

[GUI@Unidad I]\$

- startx
- .xinitrc

```
#!/bin/sh
```

```
gnome-sesion
```

```
#!/bin/sh
```

```
startkde
```

[KDE@Unidad I]\$

- KDE (K Desktop Environment) es un entorno de escritorio gráfico e infraestructura de desarrollo para sistemas Unix y, en particular, Linux.
- KDE imitó a CDE (Common Desktop Environment) en sus inicios. CDE es un entorno de escritorio utilizado por varios Unix.
- De acuerdo con su página Web, "KDE es un entorno gráfico contemporáneo para estaciones de trabajo Unix. KDE llena la necesidad de un escritorio amigable para estaciones de trabajo Unix, similar a los escritorios de MacOS o Windows".

[GNOME@Unidad I]\$

- GNOME o Gnome es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Window, se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.
- Creado por el mexicano Miguel de Icaza.

[Unidad I@presentación]#

1.1 Instalación de Linux como cliente y servidor

1.2 Instalación de software

1.3 Interfaces gráficas KDE y GNOME

1.4 Aplicaciones de uso común

1.5 Lenguajes de programación

[Ofimática@Unidad I]\$

- Ofimática: Suite con Procesador de texto, hojas electrónicas, presentaciones, gráficas y bases de datos.
- [StarOffice 6.0](#), Sun Microsystems Inc.
(Windows, Linux, Solaris)
- [OpenOffice.org](#)
(Windows, Linux, Solaris)
- Koffice

[Navegadores@Unidad I]\$

- Netscape (www.netscape.com)
- Mozilla (www.mozilla.org)
- FireFox (www.firefox.com)

- Opera (www.opera.com)
- Konqueror (KDE) (www.kde.org)

- Lynx

[Antivirus@Unidad I]\$

- OpenAntiVirus Project www.openantivirus.org
- CLAM Antivirus <http://clamav.elektropro.com>
- AMaViS – A Mail Virus Scanner www.amavis.org
- Sophos – www.sophos.com

[mtools@Unidad I]\$

- Comandos de MS-DOS
- mcopy
- /etc/mtools.conf.
- drive a: file="/dev/fd0" exclusive drive b:
file="/dev/fd1" exclusive # 1er disco Duro
drive c: file="/dev/hda1" # 2nd disco Duro
drive d: file="/dev/sda1"
mtools_lower_case=1

[Emuladores@Unidad I]\$

- DOSemu para MS-DOS
- Wine para Windows
- Virtualización:
 - VMware
 - Zen
 - Bosch

[Otras aplicaciones@Unidad I]\$

- Existen muchas aplicaciones de software libre y propietario para Linux que abarcan prácticamente todas las áreas del quehacer humano.
- La ventaja del software libre es que muchas de las versiones de programas para Linux existen también para otros SOs.

[Unidad I@presentación]#

1.1 Instalación de Linux como cliente y servidor

1.2 Instalación de software

1.3 Interfaces gráficas KDE y GNOME

1.4 Aplicaciones de uso común

1.5 Lenguajes de programación

[Lenguajes@]\$

- C Compilador gcc
 - Gcc fuente.c –o ejecutable
- C++ Compilador g++
- GNU Fortran 77

- Tcl
 - Button .b –text “Hola mundo!” –comand exit
 - Pack .b

[Lenguajes@Unidad I]\$

- Perl
- Python
- Ruby
- AWK

[Servidor Stream]@Unidad I]\$

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>

#define MYPORT 3490 // Puerto al que conectarán los usuarios

#define BACKLOG 10 // Cuántas conexiones pendientes se mantienen en cola

void sigchld_handler(int s)
{
    while(wait(NULL) > 0);
}

int main(void)
{
    int sockfd, new_fd; // Escuchar sobre sock_fd, nuevas conexiones sobre new_fd
    struct sockaddr_in my_addr; // información sobre mi dirección
    struct sockaddr_in their_addr; // información sobre la dirección del cliente
    int sin_size;
    struct sigaction sa;
    int yes=1;
```

[Servidor Stream]@Unidad I]\$

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("socket");
    exit(1);
}

if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1) {
    perror("setsockopt");
    exit(1);
}

my_addr.sin_family = AF_INET; // Ordenación de bytes de la máquina
my_addr.sin_port = htons(MYPORT); // short, Ordenación de bytes de la red
my_addr.sin_addr.s_addr = INADDR_ANY; // Rellenar con mi dirección IP
memset(&(my_addr.sin_zero), '\0', 8); // Poner a cero el resto de la estructura
}
```

[Servidor Stream]@Unidad I]\$

```
if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))
    == -1) {
    perror("bind");
    exit(1);
}
```

```
if (listen(sockfd, BACKLOG) == -1) {
    perror("listen");
    exit(1);
}
```

```
sa.sa_handler = sigchld_handler; // Eliminar procesos muertos
sigemptyset(&sa.sa_mask);
sa.sa_flags = SA_RESTART;
if (sigaction(SIGCHLD, &sa, NULL) == -1) {
    perror("sigaction");
    exit(1);
}
```

```
return 0;
```

[Servidor Stream]@Unidad I]\$

```
while(1) { // main accept() loop
sin_size = sizeof(struct sockaddr_in);
if ((new_fd = accept(sockfd, (struct sockaddr *)&their_addr,
&sin_size)) == -1) {
perror("accept");
continue;
}
printf("server: got connection from %s\n",
inet_ntoa(their_addr.sin_addr));
if (!fork()) { // Este es el proceso hijo
close(sockfd); // El hijo no necesita este descriptor
if (send(new_fd, "Hello, world!\n", 14, 0) == -1)
perror("send");
close(new_fd);
exit(0);
}
close(new_fd); // El proceso padre no lo necesita
}
```

[Cliente Stream@Unidad I]\$

- ```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
```

```
#define PORT 3490 // puerto al que vamos a conectar
```

```
#define MAXDATASIZE 100 // máximo número de bytes que se pueden leer de una vez
```

```
int main(int argc, char *argv[])
```

```
{
```

```
int sockfd, numbytes;
```

```
char buf[MAXDATASIZE];
```

```
struct hostent *he;
```

```
struct sockaddr_in their_addr; // información de la dirección de destino
```

```
memset(&(their_addr.sin_zero), 8); // poner a cero el resto de la estructura
```

# [Cliente Stream@Unidad I]\$

```
if (argc != 2) {
 fprintf(stderr, "usage: client hostname\n");
 exit(1);
}
```

```
if ((he=gethostbyname(argv[1])) == NULL) { // obtener información de máquina
 perror("gethostbyname");
 exit(1);
}
```

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
 perror("socket");
 exit(1);
}
```

```
their_addr.sin_family = AF_INET; // Ordenación de bytes de la máquina
their_addr.sin_port = htons(PORT); // short, Ordenación de bytes de la red
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
```

# [Cliente Stream@Unidad I]\$

```
if (connect(sockfd, (struct sockaddr *)&their_addr,
sizeof(struct sockaddr) == -1) {
perror("connect");
exit(1);
}
```

```
if ((numbytes=recv(sockfd, buf, MAXDATASIZE-1, 0)) == -1) {
perror("recv");
exit(1);
}
```

```
buf[numbytes] = '\0';
```

```
printf("Received: %s",buf);
```

```
close(sockfd);
```

```
return 0;
}
```

# [Servidor Datagrama@Unidad I]\$

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MYPOR 4950 // puerto al que conectarán los clientes

#define MAXBUFLEN 100

int main(void)
{
 int sockfd;
 struct sockaddr_in my_addr; // información sobre mi dirección
 struct sockaddr_in their_addr; // información sobre la dirección del cliente
 int addr_len, numbytes;
 char buf[MAXBUFLEN];

 if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
 perror("socket");
 exit(1);
 }
}
```

# [Servidor Datagrama@Unidad I]\$

```
my_addr.sin_family = AF_INET; // Ordenación de bytes de máquina
my_addr.sin_port = htons(MYPORT); // short, Ordenación de bytes de la red
my_addr.sin_addr.s_addr = INADDR_ANY; // rellenar con mi dirección IP
memset(&(my_addr.sin_zero), '\0', 8); // poner a cero el resto de la estructura
```

```
if (bind(sockfd, (struct sockaddr *)&my_addr,
sizeof(struct sockaddr)) == -1) {
perror("bind");
exit(1);
}
```

```
addr_len = sizeof(struct sockaddr);
if ((numbytes=recvfrom(sockfd,buf, MAXBUFLen-1, 0,
(struct sockaddr *)&their_addr, &addr_len)) == -1) {
perror("recvfrom");
exit(1);
}
```

```
printf("got packet from %s\n",inet_ntoa(their_addr.sin_addr));
printf("packet is %d bytes long\n",numbytes);
buf[numbytes] = '\0';
printf("packet contains \"%s\"\n",buf);
```

```
close(sockfd);
```

```
return 0;
}
```

# [Cliente Datagrama@Unidad I]\$

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define MYPORT 4950 // puerto donde vamos a conectarnos

int main(int argc, char *argv[])
{
 int sockfd;
 struct sockaddr_in their_addr; // información sobre la dirección del servidor
 struct hostent *he;
 int numbytes;

 if (argc != 3) {
 fprintf(stderr, "usage: talker hostname message\n");
 exit(1);
 }

 if ((he=gethostbyname(argv[1])) == NULL) { // obtener información de máquina
 perror("gethostbyname");
 exit(1);
 }
}
```

# [Cliente Datagrama@Unidad I]\$

```
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
 perror("socket");
 exit(1);
}
```

```
their_addr.sin_family = AF_INET; // Ordenación de bytes de máquina
their_addr.sin_port = htons(MYPORT); // short, Ordenación de bytes de la red
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
memset(&(their_addr.sin_zero), '\0', 8); // poner a cero el resto de la estructura
```

```
if ((numbytes=sendto(sockfd, argv[2], strlen(argv[2]), 0,
(struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
 perror("sendto");
 exit(1);
}
```

```
printf("sent %d bytes to %s\n", numbytes,
inet_ntoa(their_addr.sin_addr));
```

```
close(sockfd);
```

```
return 0;
}
```

# [Servidor Stream@Unidad I]\$

```
import java.net.*;
import java.io.*;

public class SocketServidor
{
 public static void main (String [] args)
 {
 new SocketServidor();
 }

 public SocketServidor()
 {
 try
 {
 ServerSocket socket = new ServerSocket (35557);

 System.out.println ("Esperando cliente");
 Socket cliente = socket.accept();
 System.out.println ("Conectado con cliente de " + cliente.getInetAddress());

 cliente.setSoLinger (true, 10);

 DataOutputStream buffer = new DataOutputStream (cliente.getOutputStream());
```

# [Servidor Stream@Unidad I]\$

```
buffer.writeInt (22);
 System.out.println ("Enviado 22");
 buffer.writeUTF ("Hola");
 System.out.println ("Enviado Hola");
```

```
DatoSocket dato = new DatoSocket();
 ObjectOutputStream bufferObjetos =
 new ObjectOutputStream (cliente.getOutputStream());
```

```
bufferObjetos.writeObject(dato);
 System.out.println ("Enviado " + dato.toString());
```

```
cliente.close();
```

```
 socket.close();
 }
 catch (Exception e)
 {
 e.printStackTrace();
 }
}
```

# [Cliente Stream@Unidad I]\$

```
import java.net.*;
import java.io.*;
```

```
public class SocketCliente
{
 public static void main (String [] args)
 {
 new SocketCliente();
 }

 public SocketCliente()
 {
 try
 {
 Socket socket = new Socket ("localhost", 35557);
 System.out.println ("conectado");
 }
 }
}
```

# [Cliente Stream@Unidad I]\$

```
DataInputStream buffer = new DataInputStream (socket.getInputStream());
```

```
System.out.println("Recibido " + buffer.readInt());
 System.out.println ("Recibido " + buffer.readUTF());
```

```
ObjectInputStream bufferObjetos =
 new ObjectInputStream (socket.getInputStream());
```

```
DatoSocket dato = (DatoSocket)bufferObjetos.readObject();
 System.out.println ("Recibido " + dato.toString());
```

```
 }
 catch (Exception e)
 {
 e.printStackTrace();
 }
}
}
```

# [Procesos@Unidad I]\$

```
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>
```

```
void trataSenhal (int);
```

```
main()
```

```
{
```

```
 pid_t idProceso;
```

```
 idProceso = fork();
```

```
 if (idProceso == -1)
```

```
 {
```

```
 perror ("No se puede lanzar proceso");
```

```
 exit (-1);
```

```
 }
```

# [Procesos@Unidad I]\$

```
if (idProceso == 0)
{
 signal (SIGUSR1, trataSenhal);
 while (1)
 pause ();
}

if (idProceso > 0)
{
 while (1)
 {
 sleep (1);
 kill (idProceso, SIGUSR1);
 }
}

void trataSenhal (int numeroSenhal)
{
 printf ("Recibida señal del padre\n");
}
```

# [Semáforos@Unidad I]\$

```
#include <iostream.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>

#if defined(__GNU_LIBRARY__) && !defined(_SEM_SEMUN_UNDEFINED)
#else
union semun
{
 int val;
 struct semid_ds *buf;
 unsigned short int *array;
 struct seminfo *__buf;
};
#endif
```

# [Semáforos@Unidad I]\$

```
main()
{
 key_t Clave;
 int Id_Semaforo;
 struct sembuf Operacion;
 union semun arg;
 int i=0;

 Clave = ftok ("/bin/lS", 33);
 if (Clave == (key_t)-1)
 {
 cout << "No puedo conseguir clave de semáforo" << endl;
 exit(0);
 }

 Id_Semaforo = semget (Clave, 10, 0600 | IPC_CREAT);
```

# [Semáforos@Unidad I]\$

```
if (Id_Semaforo == -1)
{
 cout << "No puedo crear semáforo" << endl;
 exit (0);
}

arg.val = 0;
semctl (Id_Semaforo, 0, SETVAL, &arg);

Operacion.sem_num = 0;
Operacion.sem_op = -1;
Operacion.sem_flg = 0;

while (1)
{
 cout << i << " Esperando Semáforo" << endl;
 semop (Id_Semaforo, &Operacion, 1);
 cout << i << " Salgo de Semáforo " << endl;
 cout << endl;
 i++;
}
}
```

# [RPC@Unidad I]\$

- `rpcgen -a fichero.x`

```
program NOMBRE_PROGRAMA {
 version VERSION_PROGRAMA {
 int incrementa (int) = 1;
 } = 1;
} = 0x20000001;
```

-

# [RPC@Unidad I]\$

- `int suma (int sumando1, int sumando2);`
- `struct sumandos`
  - `{`
  - `int sumando1;`
  - `int sumando2;`
  - `};`
- `program PROGRAMA_SUMA {`
  - `version VERSION_SUMA {`
    - `int suma (sumandos) = 1;`
    - `} = 1;`
  - `} = 0x20000001;`

# [RPC@Unidad I]\$

```
int * suma_1_svc(sumandos *argp, struct svc_req
 *rqstp) {
 static int result;
 /*
 * insert server code here
 */
 result = argp->sumando1 + argp-
>sumando2;
/* Esta línea debe hacerla el programador */
 return &result;
}
```

[RMli@Unidad I]\$

```
import java.rmi.*;
```

```
public interface HolaMundoRmi extends
 Remote {
 String objRemotoHola(String cliente) throws
 RemoteException;
}
```

# [RMic@Unidad I]\$

```
import java.rmi.*;

public class HolaMundoRmiC {

 public static void main(String[] args) {
 // Direccion de la maquina remota, en este caso la maquina local,
 // si se va a ejecutar en una maquina diferente, se debera cambiar
 // a algo semejante a: "rmi://www.servidor.com"
 String direccion = "rmi://10.27.34.47/";
 try {
 HolaMundoRmiI hm =
 (HolaMundoRmiI)Naming.lookup(direccion+"ObjetoHola");
 System.out.println(hm.objRemotoHola("Mundo"));
 } catch(Exception e) {
 e.printStackTrace();
 }
 System.exit(0);
 }
}
```

# [RMIs@Unidad I]\$

```
import java.rmi.*;
import java.rmi.server.*;

public class HolaMundoRmiS {

 public static void main(String args[]) {

 try {
 // Se instala el controlador de seguridad
 if(System.getSecurityManager() == null) {
 System.setSecurityManager(new RMISecurityManager());
 }

 HolaMundoRmiO objRemoto = new HolaMundoRmiO();

 Naming.rebind("ObjetoHola",objRemoto);

 System.out.println("Objeto remoto preparado");
 } catch(Exception e) {
 e.printStackTrace();
 }
 }
}
```

# [RMio@Unidad I]\$

```
import java.rmi.*;
import java.rmi.server.*;

public class HolaMundoRmiO extends UnicastRemoteObject
 implements HolaMundoRmil {

 // Constructor del objeto remoto
 public HolaMundoRmiO() throws RemoteException {
 super();
 }

 public String objRemotoHola(String cliente)
 throws RemoteException {
 return("Hola "+cliente);
 }
}
```

# [Perl@Unidad I]\$

- `#!/usr/bin/perl -w`
- `# 531-byte qrpff-fast, Keith Winstein and Marc Horowitz <sipb-iap-dvd@mit.edu>`
- `# MPEG 2 PS VOB file on stdin -> descrambled output on stdout`
- `# arguments: title key bytes in least to most-significant order`
- `$_='while(read+STDIN,$_ ,2048){$a=29;$b=73;$c=142;$t=255;@t=map{$_%16or`  
`$t^=$c^=(`
- `$m=(11,10,116,100,11,122,20,100)[$_/16%8]&110;$t^=(72,@z=(64,72,$a^=12*(`  
`$_%16`
- `-`
- `2?0:$m&17)), $b^=$_%64?12:0,@z)[$_%8]}(16..271);if((@a=unx"C*",$_)[20]&48)`  
`{ $h`
- `=5;$_=unxb24,join"",@b=map{xB8,unxb8,chr($_^$a[--`  
`$h+84])}@ARGV;s/...$/1$&/;$`
- `d=unxV,xb25,$_;$e=256|(ord$b[4])<<9|ord$b[3];$d=$d>>8^($f=$t&($d>>12^$d>>`  
`4^`
- `$d^$d/8))<<17,$e=$e>>8^($t&($g=($q=$e>>14&7^$e)^$q*8^$q<<6))<<9,$_=$t[($`  
`_]`  
`^`
- `((($h>>=8)+=$f+(~$g&$t))for@a[128..$#a])print+x"C*",@a}'s/x/pack+/g;eval`