Static configuration of network interfaces (IPv4 + IPv6)	
Commands	

Network interfaces:
Network Interfaces and addresses:
Routing tables:
Multicast addresses:
ARP/ND cache:
	`ip link`
	`ip address`
	`ip route` (`ip -6 route`, for IPv6)
	`ip maddress`
	`ip neighbour`

Network interface configuration	
`ip address add <ADDR> dev <NIC>` `ip link set dev <NIC> up`	`ip address add 10.0.0.1/24 dev eth0` `ip link set dev eth0 up`
Add a route	
`ip route add default via <ROUTER>` `ip route add <ADDR> via <ROUTER>`	`ip route add default via 10.0.0.1` `ip route add 10.0.0.0/24 via 10.0.0.1`
Examples of persistent configuration in `/etc/network/interfaces`	

`auto eth0`
`iface eth0 inet dhcp`
`auto eth1`
`iface eth1 inet static`
`  address 10.0.0.12`
`  netmask 255.255.255.0`
`  gateway 10.0.0.1`
	`auto eth2`
	`iface eth0 inet6 static`
	`  address fd00::0a:0:0:0:3`
	`  netmask 64`

Other network commands	
`nc` or `ncat`	Important options: `-l` (server mode), `-p` port
`ping`	
`ping6`	Use `-I` or `%` for zone disambiguation, when needed
`netstat`	Important options: `-t` (tcp), `-u` (udp), `-a` (all), `-n` (numeric)

Enable packet forwarding	
`sysctl net.ipv4.ip_forward=1`	`sysctl -w net.ipv6.conf.all.forwarding=1`

Dynamic configuration DHCP

1. Add DHCP server configuration in `/etc/dhcp/dhcpd.conf`. Example:
```
subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.11 10.0.0.50;
    option routers 10.0.0.3;
    option broadcast-address 10.0.0.255;
}
```
2. Start the DHCP service with `service isc-dhcp-server start` (or `stop` or `restart`)

Examples of packet filtering with `iptables`

```
iptables -P INPUT DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -s 200.1.1.1 -p tcp --dport 22 -m state --state NEW -j ACCEPT
iptables -A OUTPUT -d 22.1.1.1 -p tcp --dport 110 -m state --state NEW -j ACCEPT
```

Examples of Network Address Translation with `iptables`

```
iptables –t nat –A POSTROUTING –o ppp0 –j SNAT --to 175.20.12.1
iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE
iptables –t nat –A PREROUTING –d 175.20.12.1 –p tcp --dport 80 -j DNAT –-to 192.168.1.1:80
```

DNS	
Client tools	
`host [-v] [-t type] [name] [server]`	`host www.domain.org dnsserver.org` `host -t MX domain.org dnsserver.org`
`dig [@server] [name] [type]`	`dig www.domain.org @dnsserver.org` `dig domain.org MX @dnsserver.org`
FIle `/etc/resolv.conf`	`nameserver <IP address of DNS server>`

Example of zone definition in `/etc/bind/named.conf` and zone file

```
zone “labfdi.es.” {
  type master;
  file “/etc/bind/db.labfdi.es”;
};
```

```
$TTL 2d
$ORIGIN example.com.
example.com.  IN    SOA  ns.example.com. admin.example.com. (
                  2003080800 ; serial number
                  3h         ; refresh
                  15M        ; update retry
                  3W12h      ; expiry
                  2h20M      ; nx ttl
                  )
              IN    NS   ns
ns            IN    A    192.168.0.10
www           IN    A    192.168.0.50
```

Check syntax with `named-checkconf` and `named-checkzone`. Start the DNS service with `service bind9 start`

IPv6 prefix advertisement

1. Enable zebra daemon in `/etc/quagga/daemons` adding `zebra=yes`
2. Add the prefix to be announced by the interface in `/etc/quagga/zebra.conf`. Example:
```
interface eth0
  no ipv6 nd suppress-ra
  ipv6 nd prefix fd00:0:0:a::/64
```
3. Start Quagga with `service quagga start` (or `stop` or `restart`)

RIP configuration

1. Enable `ripd` and `zebra` daemons in `/etc/quagga/daemons`
2. Create an empty `zebra.conf` file and a `ripd.conf` file in `/etc/quagga`. Example:
```
router rip
version 2
network eth0
network eth1
```
3. Start Quagga with `service quagga start`

BGP configuration

1. Enable `bgpd` and `zebra` daemons in `/etc/quagga/daemons`
2. Create an empty `zebra.conf` file and a `bgpd.conf` file in `/etc/quagga`. Example:
```
router bgp 100
 bgp router-id 0.0.0.1
 neighbor 2001:db8:200:1::2 remote-as 200
 address-family ipv6
  network 2001:db8:100::/47
  neighbor 2001:db8:200:1::2 activate
 exit-address-family
```
3. Start Quagga with `service quagga start`

Error management	
`perror(3)`	Prints on standard error the error message associated to `errno`
`strerror(3)`	Returns the error message associated to an error code

System, user and time information	
`uname(2)`	System information (SO, host, kernel version…)
`sysconf(3)`	System limits (maximum length of arguments, path, hostname…)
`getuid(2)`, `geteuid(2)`	Real and effective user ID
`getpwnam(3)`	`passwd` file entry for a user
`time(2), gettimeofday(2)`	System time in seconds since the Epoch (1/1/1970) and in seconds and microsec.
`gmtime(3), localtime(3)`	Broken-down UTC or local time
`ctime(3)`	Convert time in seconds to a string like `"Wed Jun 30 21:49:08 1993\n"`
`strftime(3)`	Convert broken-down time to a custom string

File system management	
`open(2)`, `close(2)`	Open files and close open file descriptors
`write(2)`, `read(2)`	Read or write from an open file descriptor
`umask(2)`	Set the process's file mode creation mask, in octal or OR (|) of `S_IRUSR`, `S_IWUSR`...
`stat(2)`	Get file status. Test type (`S_ISREG()`) and permissions (`S_IXUSR`) on `st_mode`
`link(2)`, `symlink(2)`	Hard and soft link creation
`dup(2)`, `dup2(2)`	Duplicate file descriptors. Example: `int fd = open("/tmp/test", O_RDWR); dup2(fd, 1);`

`fcntl(2)`	Allows the management of advisory file locks. Example:
	`struct flock c;`
	`c.l_type = F_WRLCK; c.l_whence = SEEK_SET; c.l_start = 0; c.l_len = 0;`
	`fcntl(fd, F_GETLK, &c);`
	`if (c.l_type == F_UNLCK){ // Unlocked`
	`  c.l_type = F_WRLCK;`
	`  fcntl(fd, F_SETLKW, &c); // Lock`
	`  ...`
	`  c.l_type = F_UNLCK;`
	`  fcntl(fd, F_SETLKW, &c); // Unlock`
	`} else // Locked by c.l_pid`

`readdir(3)`	Read entries of a directory opened with `opendir(3)`

Process management	
`getpid(2)`, `getppid(2)`, `getpgid(2)`, `getsid(2)`	ID of process, parent process, process group and session
`setsid(2)`	Create a new session

`execlp(2)`, `execvp(2)`	Execute a program. Arguments as a NULL-terminated list or vector. Examples:
	`execlp("ls", "ls", "-l", NULL);`
	`execvp(args[0], args) // char *args[] = {"ls", "-l", NULL}`

`fork(2)`	Create a child process. Template:
	`pid_t pid = fork();`
	`if ( pid == -1 ) { // Error`
	`} else if (pid == 0) { // Child`
	`} else // Parent, pid is the child's PID`

`wait(2)`, `waitpid(2)`	Wait for child termination. Exit status checked with `WIFEXITED()`, `WEXITSTATUS()`...

Commands for process management	`chrt`, `nice`, `renice`, `ps`, `kill`

Functions to work with strings	`strlen`, `strcat`, `strcpy`, `strcmp`

Signal management	

`//Set signal handler for SIGTSTP`
`void handler(int signal){`
`}`
 	
`int main() {`
` struct sigaction act;`
 	
` act.sa_handler = handler;`
` act.sa_flags = 0;`
` sigaction(SIGTSTP, &act, NULL);`
` ...`
`}`
	`//Block a signal set`
	`sigset_t set, pending;`
	
	`sigemptyset(&set);`
	`sigaddset(&set,SIGINT);`
	`sigaddset(&set,SIGTSTP);`
	`sigprocmask(SIG_BLOCK, &set, NULL);`
	`...`
	`sigpending(&pending);`
	`if (sigismember(&pending, SIGINT)) {`
	`} else if (sigismember(&pending, SIGTSTP)) {`
	`}`

Pipe creation	
`pipe(2)`	Create an unnamed pipe. `fd[0]` for reading and `fd[1]` for writing. `SIGPIPE` if write to pipe with no readers
`mkfifo(3)`	Create a named pipe. Command `mkfifo` can be also used

Synchronous I/O multiplexing	

`FD_ZERO(&set);`
`FD_SET(0, &set);`
 	
`timeout.tv_sec = 2;`
`timeout.tv_usec = 0;`
 	
`rc = select(1, &set, NULL, NULL, &timeout);`
	`if (rc == -1)`
	`  perror("select()");`
	`else if (rc) { //FD_ISSET(0, &set) == 1`
	`  read(0, buffer, 80);`
	`  printf("%s\n", buffer);`
	`} else`
	`  printf("Timeout.\n");`

UDP server template (echo)	

`void main(int argc, char *argv[]) {`
` struct addrinfo hints, *res;`
` struct sockaddr_storage cli;`
` char buf[81], host[NI_MAXHOST], serv[NI_MAXSERV];`
 	
` hints.ai_flags = AI_PASSIVE;`
` hints.ai_family = AF_UNSPEC;`
` hints.ai_socktype = SOCK_DGRAM;`
` hints.ai_protocol = 0;`
` getaddrinfo(argv[1], argv[2], &hints, &res);`
` int sd = socket(res->ai_family, res->ai_socktype, 0);`
	` bind(sd,(struct sockaddr *)res->ai_addr, res->ai_addrlen);`
	` freeaddrinfo(res);`
	` while (1) {`
	`  socklen_t clen = sizeof(cli);`
	`  int c = recvfrom(sd, buf, 80, 0,`
	`  (struct sockaddr *) &cli, &clen);`
	`  getnameinfo((struct sockaddr *) &cli, clen, host,`
	`  NI_MAXHOST, serv, NI_MAXSERV, NI_NUMERICHOST);`
	`  printf("%s %s\n", host, serv);`
	`  sendto(sd, buf, c, 0, (struct sockaddr *)&cli, clen);`
	` }`
	`}`

TCP server template (echo)	

`void main(int argc, char *argv[]) {`
` struct addrinfo hints, *res;`
` struct sockaddr_storage cli;`
` char buf[81];`
 	
` hints.ai_flags = AI_PASSIVE;`
` hints.ai_family = AF_UNSPEC;`
` hints.ai_socktype = SOCK_STREAM;`
` hints.ai_protocol = 0;`
` getaddrinfo(argv[1], argv[2], &hints, &res);`
` int sd = socket(res->ai_family, res->ai_socktype, 0);`
	` bind(sd,(struct sockaddr *)res->ai_addr, res->ai_addrlen);`
	` freeaddrinfo(res);`
	` listen(sd, 5);`
	` while (1) {`
	`  socklen_t clen = sizeof(cli);`
	`  int cli_sd = accept(sd, (struct sockaddr *) &cli, &clen);`
	`  int c = recv(cli_sd, buf, 80, 0);`
	`  send(cli_sd, buf, c, 0);`
	`  close(cli_sd);`
	` }`
	`}`