**RC**

**Seminar I**

ftp – client (try on cmd)

FTP – protocol

tracert [www.google.com](http://www.google.com) – arată câte noduri se fac de la sursa pana la destinatie (try on cmd)

ping [www.google.com](http://www.google.com) – arată pachetele (trimise, primite si pierdute) (try on cmd)

32 de poziții hexa => 32\*4 = 128 biți în hexa

Pt a avea acces la DHCP server avem nevoie ca serviciul DHCP client sa fie running

De identificat adresa ip (daca e locala sau externa ) (daca e permanenta sau temporala) (de indetificat dispozitivul care ofera acesta adresa de i ?????? routerul

De uitat la router

De invatat porturile din client/server din s1-s2-servicii-internet-n.doc

**Seminar II**

**Port** = e o resursă software și e alocată unic pt fiecare aplicație de tip server. Dacă avem mai multe servere cu același port, primul server care va porni va primi resursa (portul), celelalte nefuncționând. Portul e un număr pe 16 biți.

*65535 – nr maxim de porturi care pot fi la un pc*

*Primele 1024 de porturi aparțin SO.*

*Servere web:*

**IIS** – poate avea mai multe servere cu același nume.

**HTTPD** – d vine de la daemon, și e un program de Unix care stă și lucrează în background

**APACHE** – cel mai răspandit server existent astăzi.

**carment.ase.ro --> rc-> sem->s1 -> Serviciul Web --> URL**

protocol://server.subdom.dom:[port]/dir/subdirt/denfis.htm 🡺 **Schemă URL**

80 port 1079 ==> 8079

443 => 44379

**Seminar III**

**Folderele Apache24:**

**Bin** – fisierele executabile (vom folosi .*httpd.exe, .htpasswd.exe*)

Cu primul vom lansa apachul cu diferite optiuni

Cu al doilea vom crea parole

**Conf** – folder pt fisierele de configurare. Principalul fisier de configurare e *httpd.conf*

**Conf extra** – fisierele extra de configurare

**Htdocs** – director care contine fisierele html pe care le serveste serverul. Folderul din care pornesc paginile de web. Pagina implicită de pornire este: *index.html*

**Logs** – se creeaza fisierele de acces la servere (*error.log, httpd.pid*)

**Modules** – toate modulele pe care le putem utiliza in utilizarea serverului (apar din compilare)

<http://httpd.apache.org/docs/2.4/>

<http://httpd.apache.org/docs/2.4/mod/directives.html>

***Directive de configurare:***

**ServerRoot** – definește punctul de intrare in sistem, de unde își încarcă directivele de configurare, error, etc. (este case senstive)

**Listen 80** – stabileste portul pe care va porni serverul. 80 e portul implicit

**User daemon**

**Group daemon** - arata sub ce user si group va fi pornit serverul

**ServerAdmin** [**admin@example.com**](mailto:admin@example.com) – stabileste adresa de mail a administratorului serverului

**ServerName** [**www.example.com:80**](http://www.example.com:80) – stabileste care e numele serverului si portul pe care va merge

**<Directory />**

**AllowOverride none**

**Require all denied**

**</Directory>** – directiva pereche, stabileste drepturile de acces la directoare. Are multe optiuni

**DocumentRoot "${SRVROOT}/htdocs"** – reprezinta directorul din care vor fi prezentate fisierele html

**DirectoryIndex index.html** – stabileste pagina web de pornire a serverului

**<Files ".ht\*">**

**Require all denied**

**</Files>** - directiva pereche, stabileste drepturile de acces la fisiere

**ErrorLog "logs/error.log"** – stabilește locația și numele fișierului de error

**CustomLog "logs/access.log" common** – e directiva care stabileste care e directorul si numele fișierului de acces la server. Common reprezinta modul de acces. Fisierul e *access.log*

**Seminar IV**

**Userdir.conf** – directiva se gaseste in modul userdir trebuie setata cu un anumit director

**Seminar V**

**.htaccess, .htpasswd** – fisiere de securitate ale apache-ului

**Seminar VI**

**Server** – avem nevoie de port și pidfile unice

**Server virtual** – am mers pe același port și pidfile ca la serverul virtual (pot fi n virtual hotspots)

**Host virtual** – poate fi accesat si din afara rețelei

**Director virtual** – asociem un director (cu tilda) ??

pwd – arată calea

/home/an3/gr5\_8 – calea mea

-=fis,

d=dir,

l sau @=link

rwx – drepturi unix sunt pt grup, other, u = > toate formleaza a (all)

r – read (minim)

w – write (doar pt user trebuie sa existe)

x – executie

pag web a unui user tb sa : r-x pt gr5\_..., dir UserDir, continut directorpt o (other)

**chmod -R** – ofera drepturi

111=7 (rwx=user); 5=101 (r-x=pt grup); 5=101 (r-x=pt other)chmod -R 755 dir

cat > fis5 – creezi fisier

ps – pt a afla pidul

**pstree --?**

netstat -a -n -p|grep -w 80 - vedem daca portul 80 e ocupat

whereis httpd – vedem calea apache-ului

**Seminar VIII**

http://carment.ase.ro/licenta/rc/semhttp://carment.ase.ro/licenta/rc/sem/socket/socket-doc/thunder.ocis.temple.edu/netprog5.html#sockets\_concepts

http://beej.us/guide/bgnet/html/

Socket – interfață standard, implementată de kernel-ul SO

Protocoalele TPC/IP de nivel transport oferă servicii ce permit programelor nivelului aplicaţie să comunice între ele prin intermediul mesajelor;

Rând o aplicaţie trimite o cerere către nivelul transport  pentru a trimite un mesaj, protocolul folosit la acest nivel:

o    împarte informaţia în pachete

o   adaugă un antet de pachet care include adresa destinaţiei

* trimite informaţia nivelului reţea pentru procesare ulterioară

Transmisia şi recepţia datelor se realizează prin intermediul unor porturi de pe server, care identifică destinaţia specifică a mesajului;

Nivelul transport este implementat în reţelele TCP/IP prin intermediul a două protocoale:

o   **UDP (User Datagram Protocol)**protocol datagramă utilizator 

  Asigură *servicii de tip datagramă*nivelului aplicaţie;

  *Nu este fiabil* (nu asigură certitudinea livrării datagramelor, nici mecanismele de protecţie la pierderea sau duplicarea datagramelor);

  *Viteză mare de transmisie*;

  Este un *serviciu fără conexiune* (emiţătorul nu cunoaşte starea receptorului în momentul transmisiei);

  Pt. transferul datelor foloseşte nişte entităţi abstracte, numite porturi de protocol, identificate prin numere întregi pozitive şi care au asociate nişte cozi de mesaje prin care se transmit mesajele;

  Se utilizează pt. mesaje mici (sub 8KB) cu viteză mare;

  Antetul datagramei UDP conţine:

       Source Port Number- adresa portului sursă;

       Destination Port Number  adresa portului destinaţie;

       Length  lunginea datagramei în bytes;

       Checksum  suma de control asociată datagramei (foloseşte acelaşi algoritm ca la protocolul IP)

o   **TCP (Transmission Control Protocol)-** protocol de control al transmisiei

  Este fiabil (asigură integritatea datelor transmise, mecanisme de protecţie la pierderea sau duplicarea pachetelor, păstrarea numărului de secvenţă, mecanisme de control al fluxului de date în reţea);

  Asigură transmisia blocurilor continue de date între porturile de protocol asociate aplicaţiilor;

  Dimensiunea mesajelor nu este limitată;

  Viteza de transfer mai mică;

SO oferă programelor la nivel aplicaţie o interfaţă comună pt. aceste 2 protocoale, şi anume interfaţa ***socket***.

**2.    Interfaţa Socket API**

Este o interfaţă între un program de aplicaţie şi serviciul de transport (este un standard de facto), fiind furnizat de o bibliotecă socket sau de sistemul de operare. Se foloseşte conceptul de descriptor, fiecare socket fiind tratat asemănător cu un fişier local. Acest descriptor este transmis aplicaţiei la crearea socket-ului şi apoi este utilizat ca argument în apelurile următoare

**Primitive de serviciu Socket API**

|  |  |
| --- | --- |
| Descriptor = socket(protofamily, type, protocol) | creează un socket |
| close(socket) | închide un socket |
| bind(socket, localaddr, addrlen) | leagă socket-ul cu un port |
| listen(socket,queuesize) | pune socket în mod pasiv |
| newsock = accept(socket, caddress, caddresslen) | acceptă o cerere de conectare |
| connect(socket, saddress, saddresslen) | stabileşte legătura cu un server care a făcut accept |
| send(socket, data, length, flags) | transmite un mesaj |
| sendto(socket, length, flags, destaddress, addresslen) | transmite un mesaj folosind un socket neconectat |
| sendmsg(socket, msgstruct, flags) |  |
| recv(socket, buffer, length, flags) | primeşte un mesaj |
| recvfrom(socket, buffer, length, flags, sndaddr, saddrlen) | primeşte un mesaj pe un socket neconectat |
| rcvmsg(socket, msgstruct, flags) |  |

AF- adress family

AF\_INET – Ipv4 Internet protocols

SOCKET(2) Linux Programmer's Manual

NAME

socket - create an endpoint for communication

SYNOPSIS

#include <sys/types.h> /\* See NOTES \*/

#include <sys/socket.h>

int socket(int domain, int type, int protocol);

DESCRIPTION

socket() creates an endpoint for communication and returns a file des

to that endpoint. The file descriptor returned by a successful call wi

numbered file descriptor not currently open for the process.

The domain argument specifies a communication domain; this selects

which will be used for communication. These families are defined in <s

currently understood formats include:

Name Purpose Man page

AF\_UNIX, AF\_LOCAL Local communication unix(7)

AF\_INET IPv4 Internet protocols ip(7)

AF\_INET6 IPv6 Internet protocols ipv6(7)

AF\_IPX IPX - Novell protocols

AF\_NETLINK Kernel user interface device netlink(7)

AF\_X25 ITU-T X.25 / ISO-8208 protocol x25(7)

AF\_AX25 Amateur radio AX.25 protocol

AF\_ATMPVC Access to raw ATM PVCs

AF\_APPLETALK AppleTalk ddp(7)

AF\_PACKET Low level packet interface packet(7)

AF\_ALG Interface to kernel crypto API

The socket has the indicated type, which specifies the communication se

defined types are:

SOCK\_STREAM Provides sequenced, reliable, two-way, connection-based

out-of-band data transmission mechanism may be supporte

SOCK\_DGRAM Supports datagrams (connectionless, unreliable messages

length).

SOCK\_SEQPACKET Provides a sequenced, reliable, two-way connection-base

path for datagrams of fixed maximum length; a consumer

an entire packet with each input system call.

SOCK\_RAW Provides raw network protocol access.

SOCK\_RDM Provides a reliable datagram layer that does not guaran

SOCK\_PACKET Obsolete and should not be used in new programs; see pa

Some socket types may not be implemented by all protocol families.

The protocol specifies a particular protocol to be used with the socke

single protocol exists to support a particular socket type within a giv

in which case protocol can be specified as 0. However, it is possible

may exist, in which case a particular protocol must be specified in thi

tocol number to use is specific to the “communication domain” in which

take place; see protocols(5). See getprotoent(3) on how to map protoco

protocol numbers.

The AF\_ALG protocol type was added in Linux 2.6.38. More information o

provided with the kernel HTML documentation at https://www.k

docs/crypto-API/User.html.

cu socket lucreaza:

accept(2), bind(2), close(2), connect(2), fcntl(2), getpeername(2), ge

sockopt(2), ioctl(2), listen(2), read(2), recv(2), select(2), send(2),

etpair(2), write(2), getprotoent(3), ip(7), socket(7), tcp(7), udp(7),

<https://www.kernel.org/doc/man-pages/>

4.2.3 System Calls, Library Calls, Header Files, and Data Structures – de stiut pt data viitoare