96 Overheads

Part 1: Networking basic concepts, DNS

More about this course about Internet application protocols can be found at URL:

http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html

Last update: 05-01-14 18.29

Important information about this course segment

Lectures are not mandatory

But there can be questions in the exam on what is said during the lecture. Reading the written material carefully, and trying to understand or find out the ideas behind the overhead slides in the compendiums, will give you the necessary information to pass the exam.

Lectures may not exactly follow the lecture schedule, and I may skip some things in the end.

Requirements

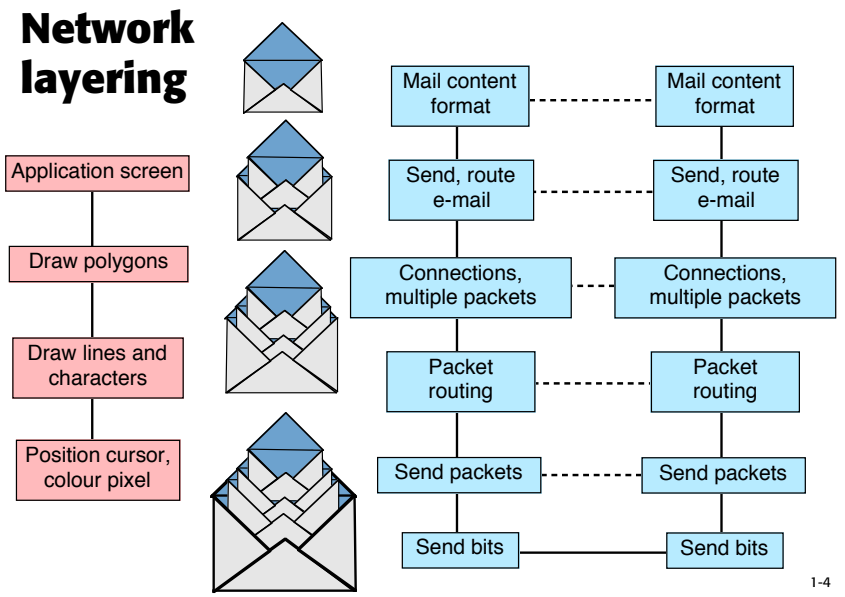
Exam: Some of the compendiums are allowed during the exam, others are not. This is marked on the front page of the compendium. Even though some exams may be marked for KTH or for SU, any student can go to any exam, provided that you notify in advance.

Work task: Prepare an XML DTD and an XML code using this DTD. Check them against an XML validator. See course description for more info.

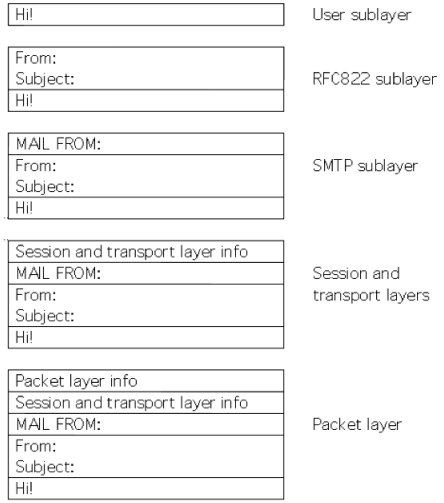
Prio: When an item in the course schedule is marked Prio, this means that certain computer rooms are booked for work on the work task. You can go to computer rooms at other times, if there are seats available. No supervisor will help you with the work task during these periods.

Mailing list

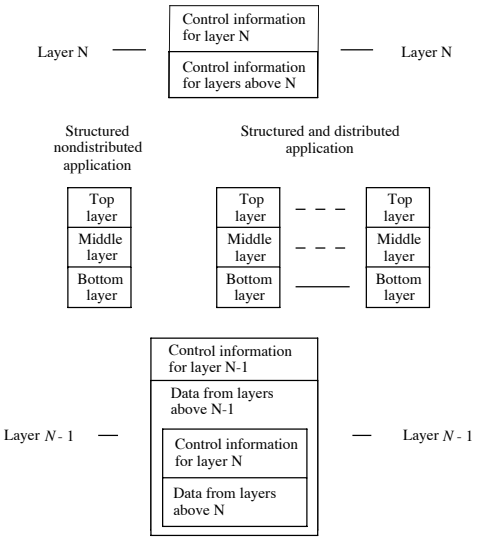
Either participate in the First Class conference for this course, or subscribe to the mailing list.



Layering example



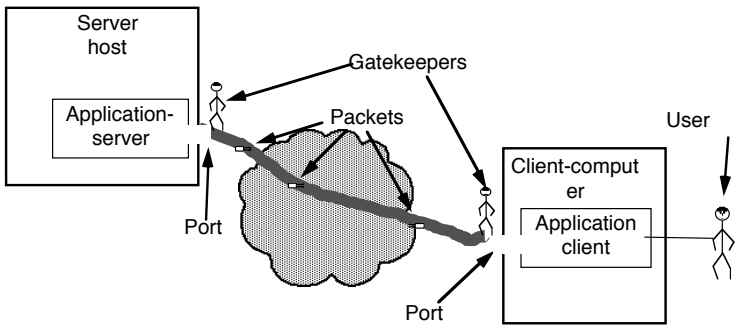
Understanding layering



Overview of Internet protocols and services

Protocol name	Main usage	Clients	Servers
DNS	Translating domain names to numerical host addresses	All kinds of clients and name servers	Name servers
HTTP (and HTML)	Downloading web pages in the WWW. Can also be used to send in filled in forms and to send in files. Also used for many specialized protocols based on HTTP.	Web browsers	HTTP servers
SMTP (and RFC822 and MIME)	Sending and forwarding of e-mail to and between MTAs (Message Transfer Agents)	Mail clients and SMTP servers	SMTP servers
POP and IMAP	Downloading of e-mail to the mail clients of their recipients	Mail clients	POP or IMAP servers
NNTP	Downloading and forwarding of Usenet News articles.	News clients and news servers	News servers
FTP	Anonymous downloading of files, non-anonymous transfer of files between logged in directories.	FTP clients, Web browsers	FTP servers
Gopher	An old, nowadays not much used protocols, which can be seen as a limited subset of HTTP.	Web browsers, Gopher clients	Gopher servers
PICS	”Protection” of children from material on the net regarded as unsuitable for them.	All kinds of clients	PICS servers
LDAP	Searching in directories.	LDAP clients, often built into e-mail clients.	LDAP servers

Computers, applications, ports, packets



One host can have many different ports fpr different applications.
Exampels of ports: E-mail, file transfer, World Wide Web.
All communication to one particular port uses one particular language.

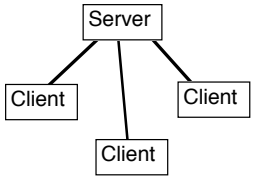
Registered port numbers

Port numbers are registered with IANA (Internet Assigned Numbers Authority)

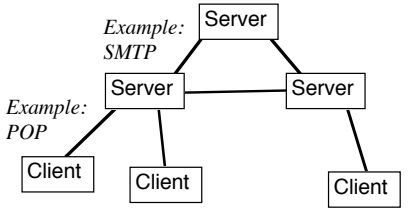
The port numbers are divided into three ranges: the Well Known Ports, the Registered Ports, and the Dynamic and/or Private Ports.
The Well Known Ports are those from 0 through 1023.
The Registered Ports are those from 1024 through 49151.
The Dynamic and/or Private Ports are those from 49152 through 65535.

```
ftp-data     20  File Transfer [Default Data]
ftp          21  File Transfer [Control]
telnet       23  Telnet
smtp         25  Simple Mail Transfer
nameserver   42  Host Name Server
nicname      43  Who Is
domain       53  Domain Name Server
whois++      63  whois++
gopher       70  Gopher
finger       79  Finger
http         80  World Wide Web HTTP
www-http     80  World Wide Web HTTP
kerberos     88  Kerberos
hostname    101  NIC Host Name Server
pop2        109  Post Office Protocol - Version 2
pop3        110  Post Office Protocol - Version 3
sunrpc      111  SUN Remote Procedure Call
auth        113  Authentication Service
uucp-path   117  UUCP Path Service
nntp        119  Network News Transfer Protocol
imap2       143  Interim Mail Access Protocol v2
imap3       220  Interactive Mail Access Prot. v3
```

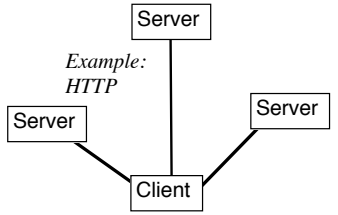

Architectures



Example: LAN data base

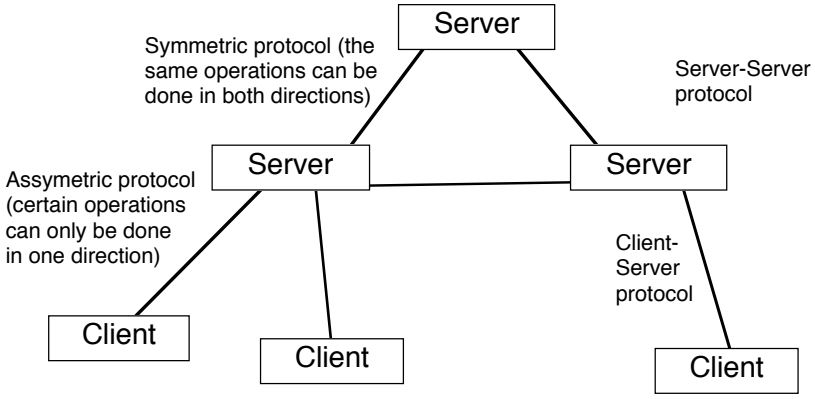


Example: E-mail, Usenet News

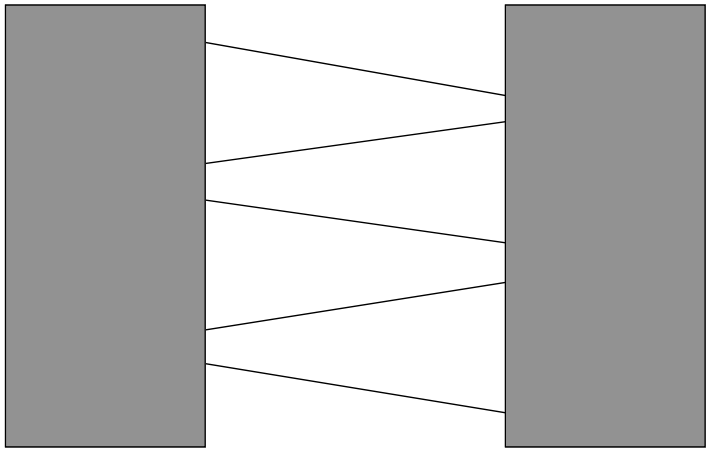


Example: WWW

Symmetric and asymmetric protocols

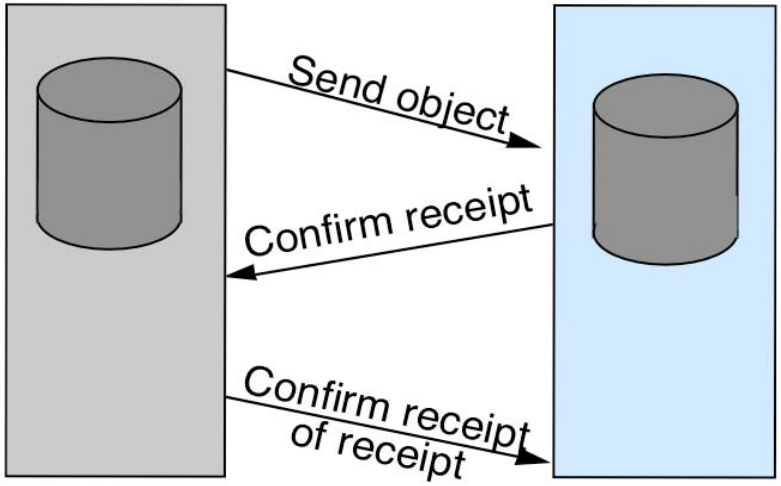


Protocols

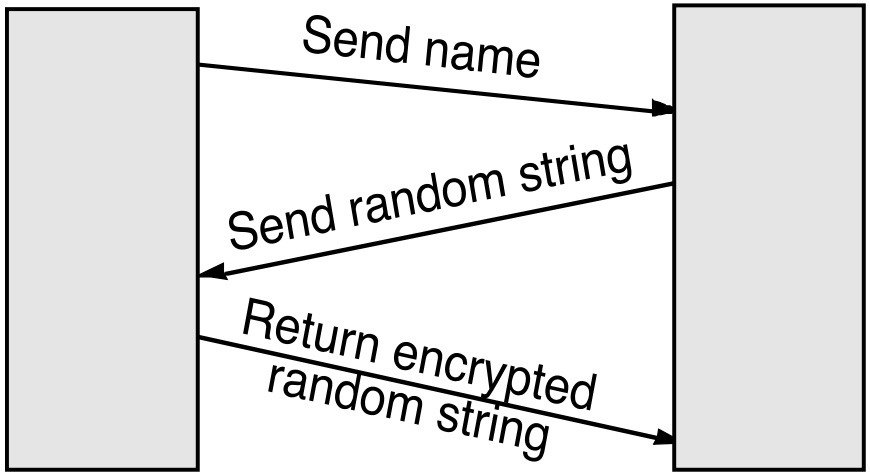


Confirmations, error codes, responses

Transfer of responsibility



Identification



Public/secret key encryption

encrypted text = f_1(original text)
original text = f_2(encrypted text)
Can f_2 be derived from f_1?

Pros and cons of public key encryption

+ Solves partly key transportation problem
– More CPU-time consuming

Authentication, authorization

- To verify the sender of a message
- Payments, agreements
- UA-UA or MTA-MTA

Authentication methods

(a) Passwords

(b) Specially designed networks

(c) Public key cryptography

Three levels of protection of message transmission:

(1) The agents identify each other using noninvertible forms of ordinary passwords. This is called *weak authentication*.

(2) The agents identify each other using public key encryption algorithms. This is called *strong authentication*.

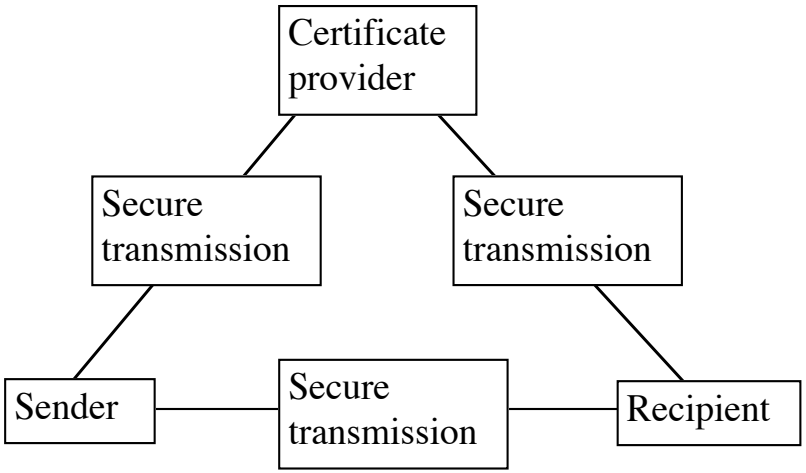
(3) Strong authentication is combined with encryption of all messages during the whole transmission.

Digital Signatures and Digital Seals

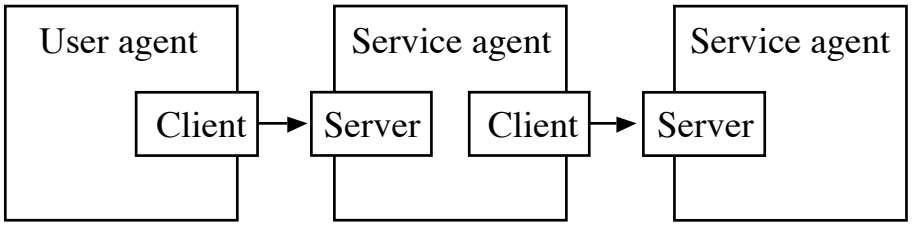
Methods: Secret key encryption of signature or checksum, which anyone can decrypt with public key

- Number of interactions
- Need of a neutral third party
- Bilateral or open to groups

Certificate Authorities

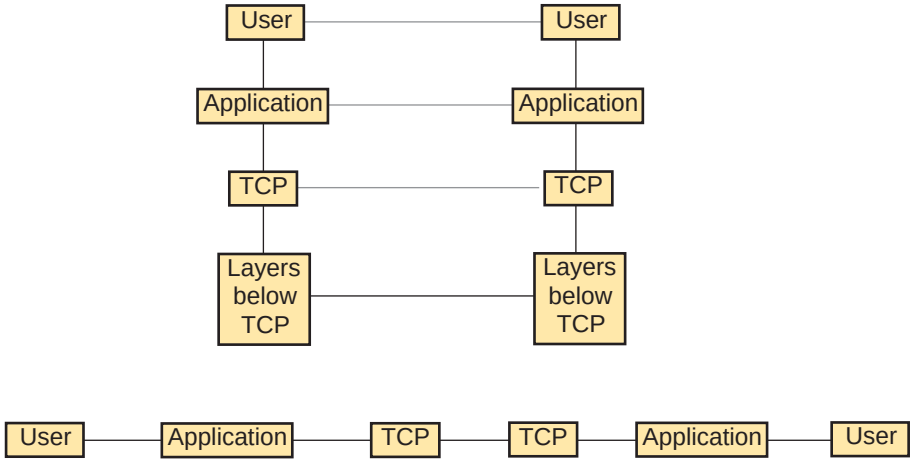


Store-and-forward transmission



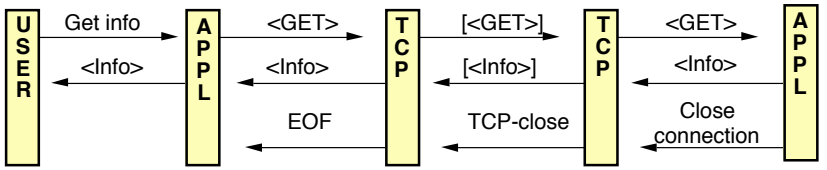
3-21

Ending a connection 1: notation to be used

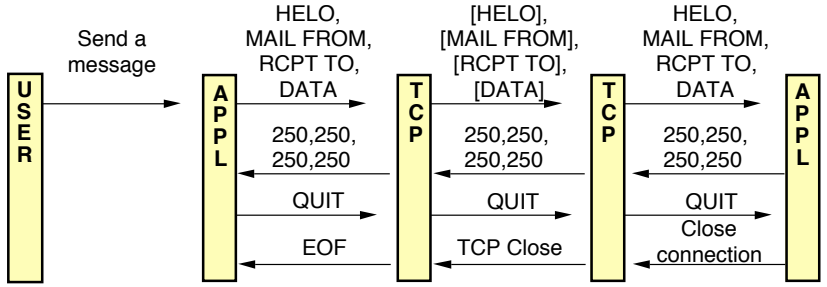


3-22

HTTP GET Operation



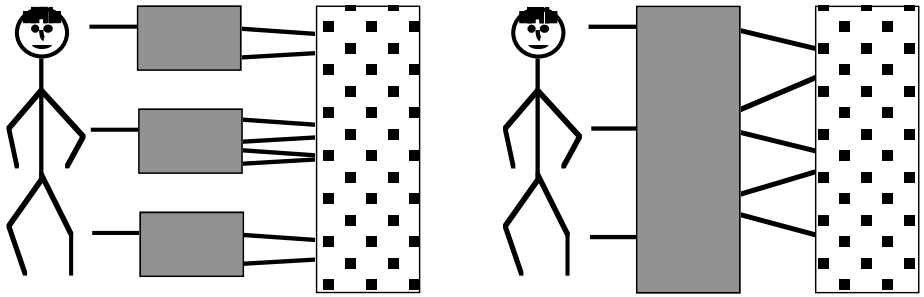
SMTP Sending a message



3-23

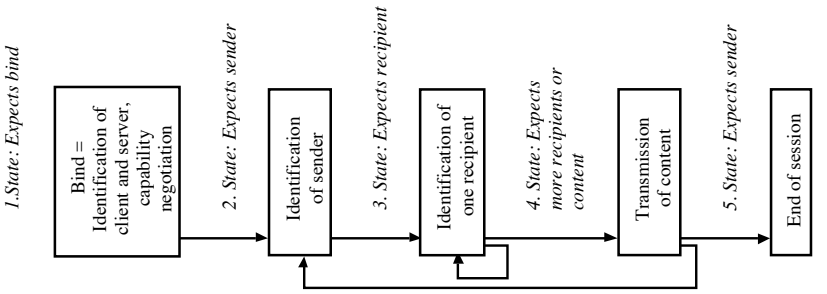
Connection retention

Transaction processing versus connection-oriented protocols



3-24

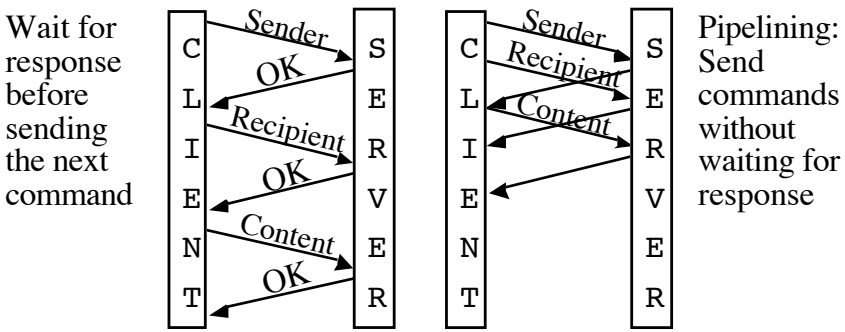
Stateful and stateless sessions



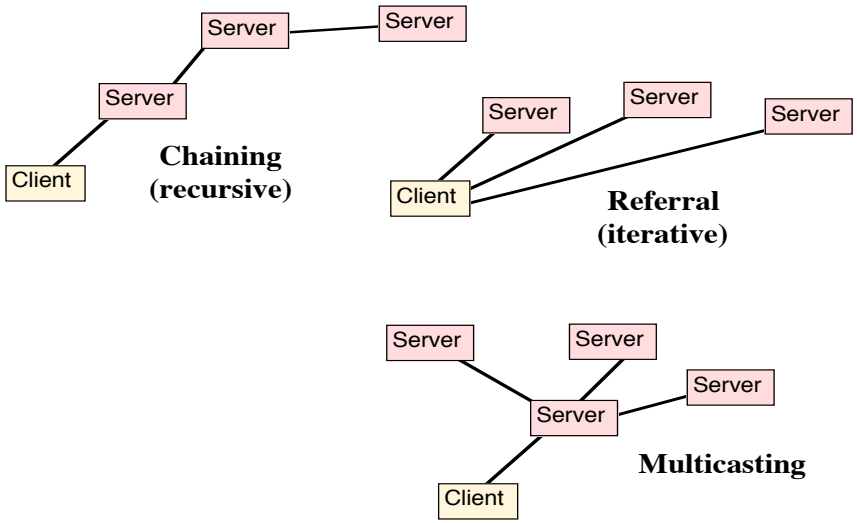
Reducing turn-around time

1. *Specify more powerful commands*, where more is done in one command, so that fewer interactions are needed.
2. *Open several parallel connections.* HTTP clients (web browsers) often keep four parallel connections for downloading the different parts of a web page (text, pictures, applets). Too many parallel connections is costly in resources for both the client and server, but with too few connections, dead time may occur when the client is waiting for data from all the connections.
3. In protoocols which use many small interactions, such as SMTP and NNTP, the delay can be used with *pipelining*, see next overhead.

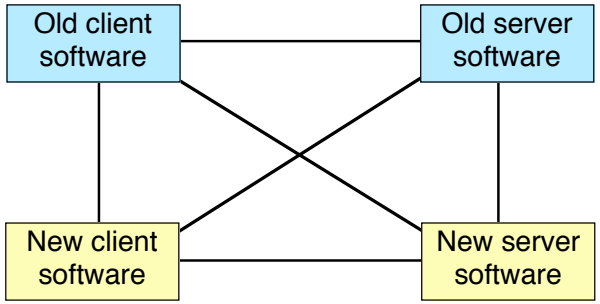
Pipelining



Chaining, referral, multicasting

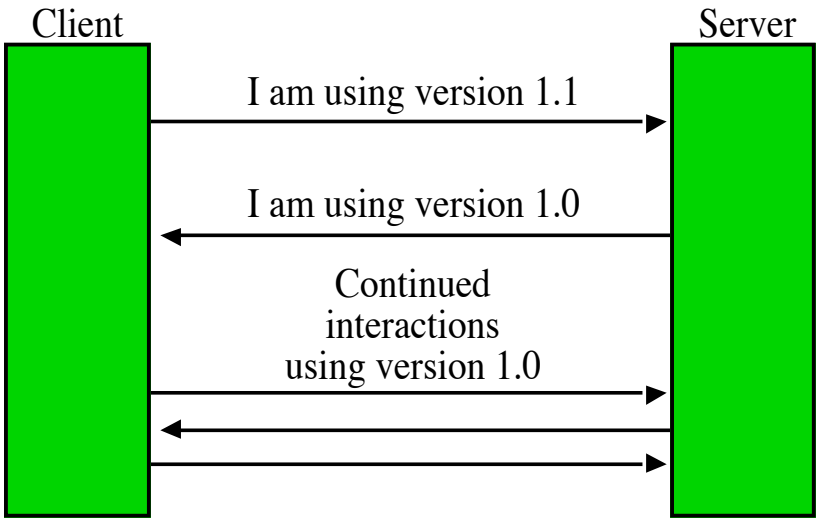


Extension problem

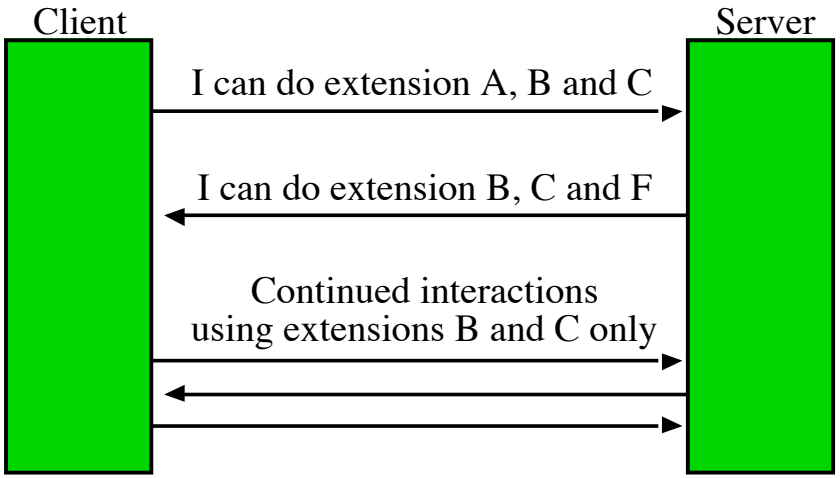


Horror example: Binary files through 7-bit e-mail
Extension by levels: for example HTML 1.0, HTML 2.0
Extension by feature selection
Built-in extension points
Registration facility vers. X-headers

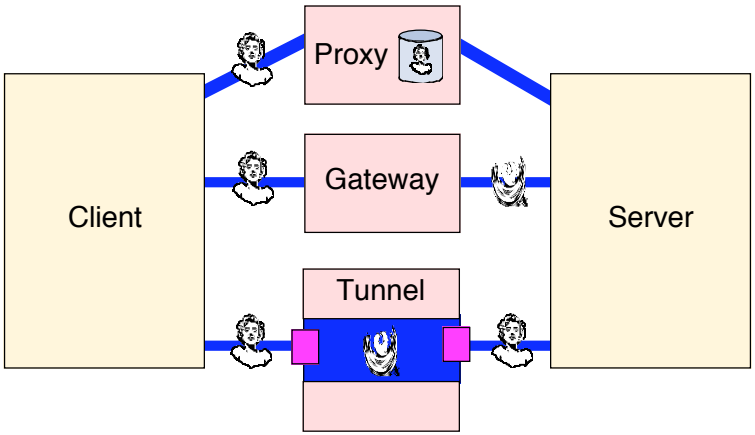
Version number method



Feature selection method



Intermediaries

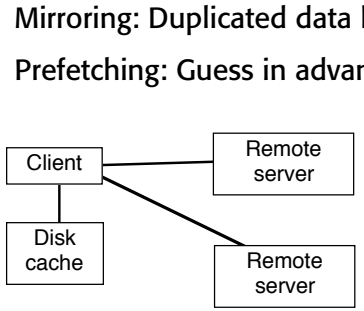


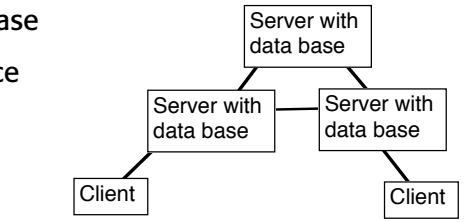
Caching: Saving a copy

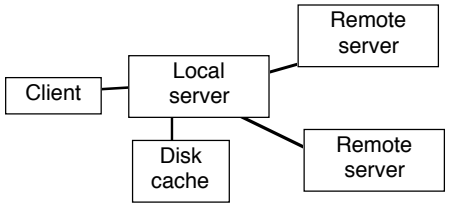
Shadowing (push-caching): Agreed, controlled replication

Mirroring: Duplicated data base

Prefetching: Guess in advance







Replication

Why replication

Reduce network load

Reduce load on very popular servers

Example: Popular home page required nine dedicated servers and rotating DNS server to distribute load

Faster response times

Master versus equalitarian replication

One master copy: All other instances are copies of the master

Replication can be pre-ordered, automatic, initiated by the

Example: Most shareware data bases

No master copy: All instances are equal

Example: Usenet News

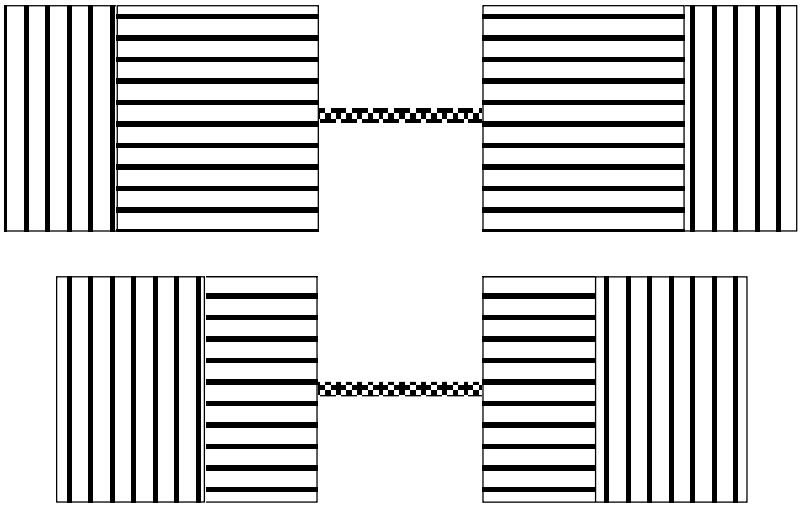
Pros and cons

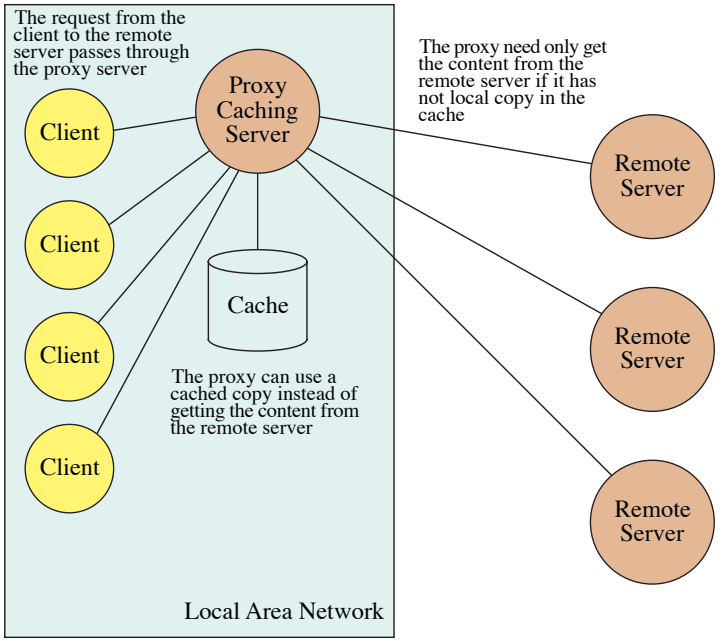
Master copy gives simpler and safer updating

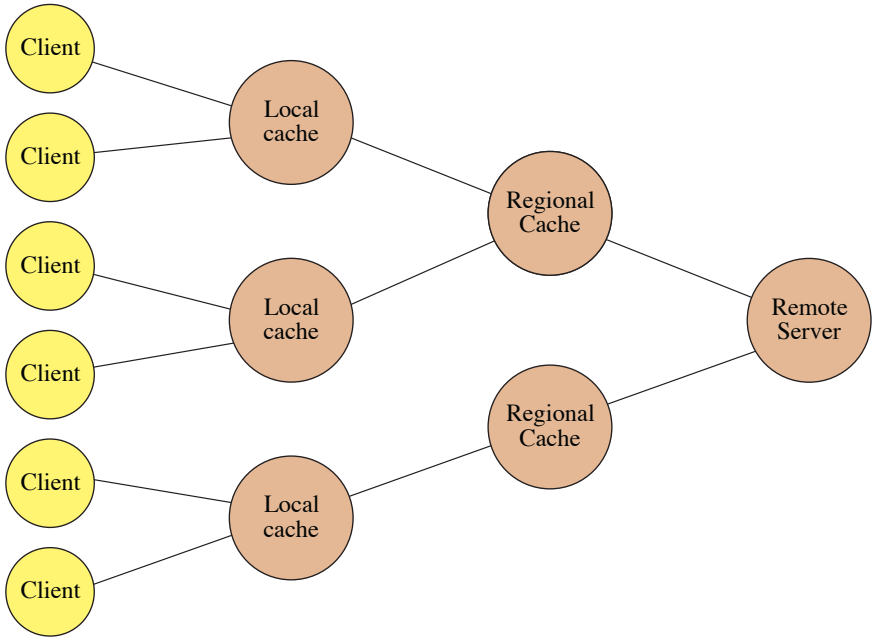
Master copy gives central control - note virus control!

Master copy depends on the master server

Replicate much or little?



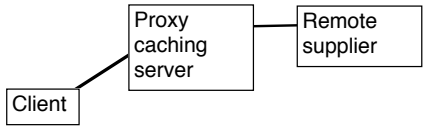




Negative caching

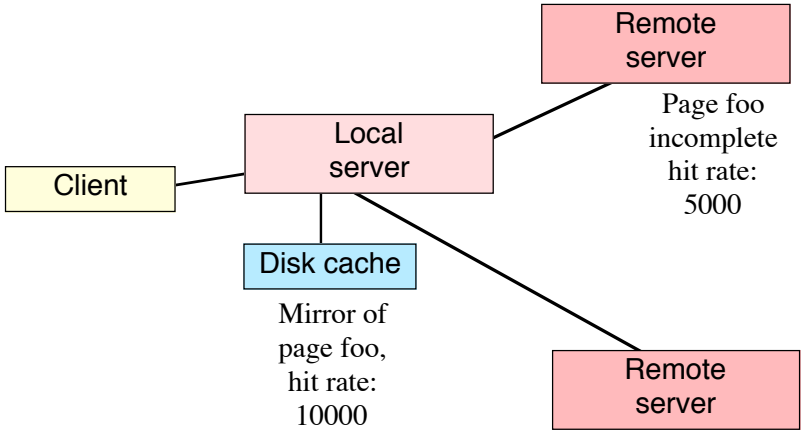
Caching of the fact that something does not exist, to avoid trying to get it several times from a remote source

Problems with caching



- User gets out-of-date result
  - Example: Cartoon changed daily, proxy caching server updated graphics only every 14 days!
- Copyright violation?
- Solutions:
  - Best solution: Controlled mirroring
  - Cacher checks for changes
  - Provider supplies refresh time
  - Guess at refresh based on last update
  - User pushes refresh button
    - Important: User refresh request must go all the way

Access statistics not correct



Security and caching

IP authentication defeated - maybe an advantage?

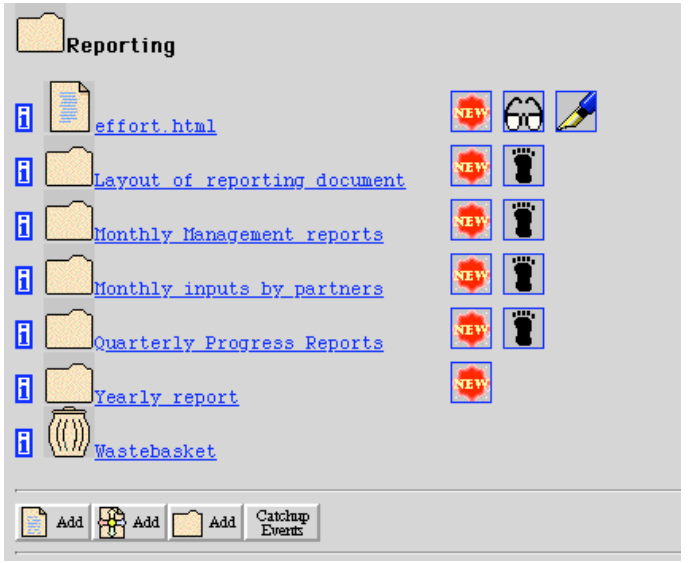
Crypthographic encryption and authentication will work

Cache manager of course a security risk

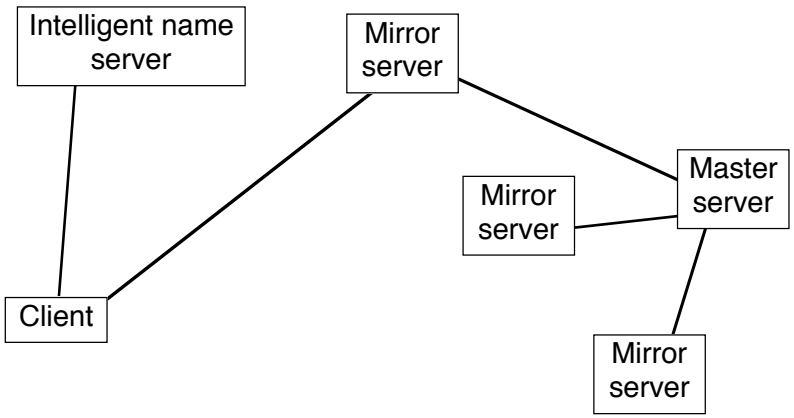
Cache must be programmed not to support private pages to non-authorised raders

Designing web page to utilize caching

Many small graphics, reuse the same graphic several times



Locating nearest copy



IETF Standards terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Classes of standards

- Experimental standard
- Proposed standard
- Draft standard
- Standard
- Historical
- Informational
- BCP

1. MUST This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

2. MUST NOT This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

3. SHOULD This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. SHOULD NOT This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

5. MAY This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

The First Golden Rule:

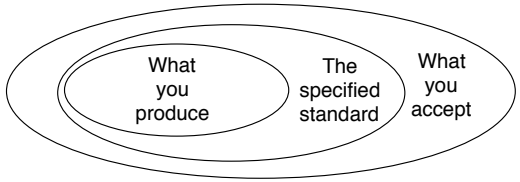
Be liberal in what you accept,
be conservative in what you pro

Does this mean a different protocol and syntax for what you produce and what you accept?

How do you know what (in excess to the standard) you should accept, and what (included in the standard you should not produce)?

Example; e-mail: Do not use blanks in e-mail names

Example; e-mail: Accept
John T. Smith <jsmith@foo.bar.net>



Golden Rules

(1) Be liberal in what you accept, be conservative in what you produce

Use a narrow produce syntax and a wide accept syntax

(2) Do no harm

What may be good in your special case, may in other cases cause harm

(3) Do not munge

Munge = Modify what other network modules has produced

Names in the Internet

Physical net addresses, example: 130.237.161.10

Domain names, example: ester.dsv.su.se, eies2.njit.edu

E-mail-addresses: example: president@whitehouse.gov

DNS = Domain Naming Service translates domain names to physical net addresses. Can be accesses through the client "nslookup" (RFC 1034, RFC 1035)

People seldom see the physical net addresses, since translation from domain names to physical net addresses is done by the application programs used.

Domain naming tree structure

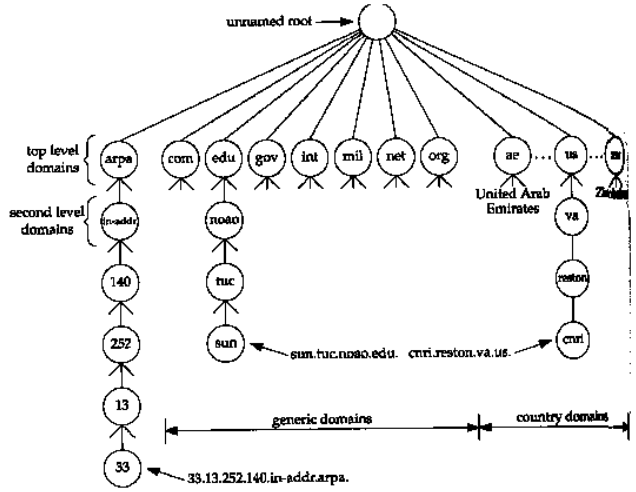


Figure 14.1 Hierarchical organization of the DNS.

Picture from Stevens: TCP/IP Illustrated, Volume 1 page 188

Top level domains

Organizational groups (U.S. or international)

COM	for commercial companies
EDU	for schools and universities
GOV	for other government agencies
INT	for international and multinational organizations
MIL	for military organizations
NET	network providers and gateways
ORG	for organizations

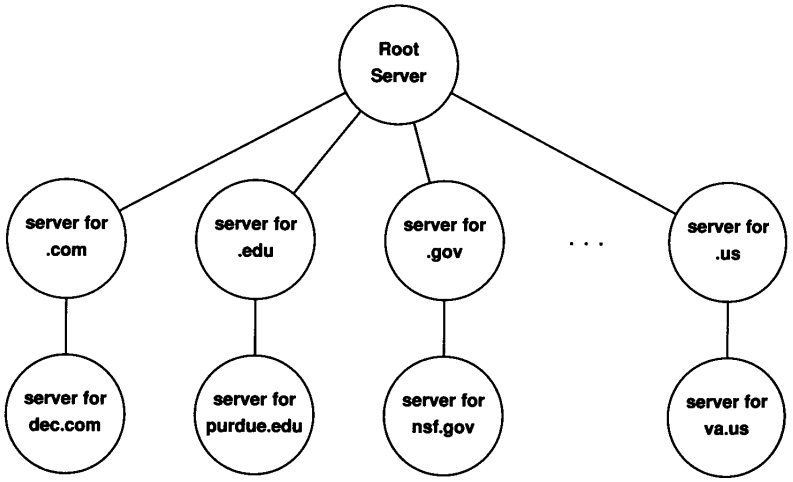
New TLDs?

FIRM	Businesses, firms
STORE	Offering goods to purchase
WEB	Activities related to the WWW
ARTS	Cultural and entertainment
REC	Recreational/entertainment
INFO	Information services
NOM	Personal names, etc.

Countries

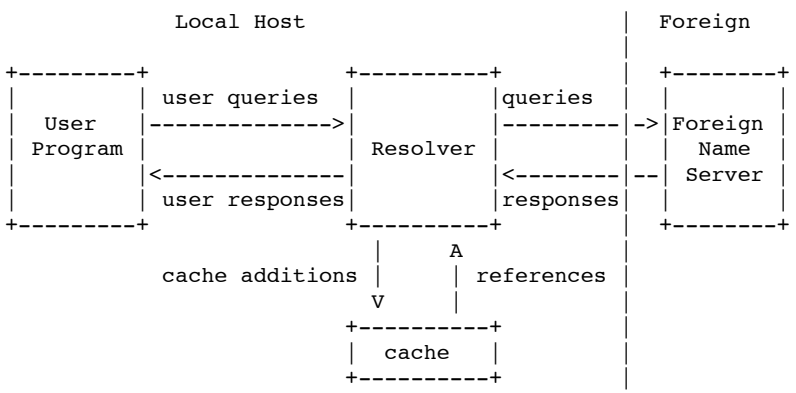
US	U.S.A.
UK	for United Kingdom
SE	for Sweden
FR	for France
etc.	

Conceptual arrangement of name server hiearchy



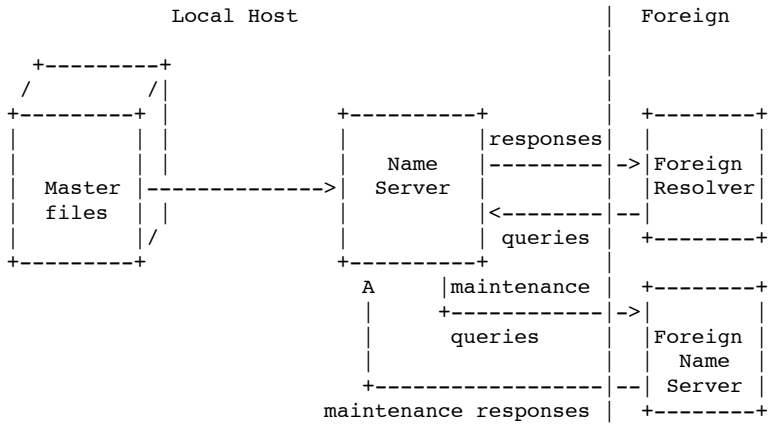
Picture from Comer: Internetworking with CTP/IP, Volume 1, page 392

Architecture: Simplest variant



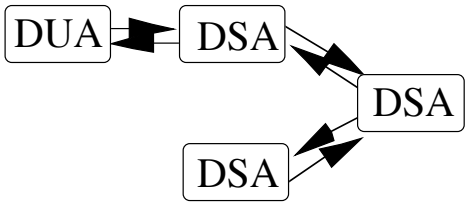
Picture from RFC 1035

Architecture: Maintenance

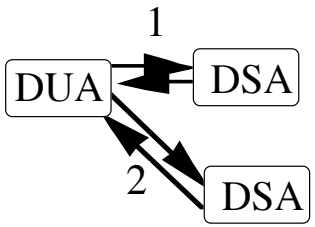


Picture from RFC 1035

Use of multiple name servers



Chaining (recursive look up)



Referral (iterative look up)

Resource Records

A	IP address
PTR queries	Pointer record used for pointer (address to domain)
CNAME	Canonical name (for an alias)
HINFO	CPU and operating system of host
MX	Mail exchange record

MX records preference value

```
host -tv mx dsv.su.se
mars.dsv.su.se    86400 IN    MX    0
jupiter.dsv.su.se 86400 IN    MX    10
sunic.sunet.se    86400 IN    MX    20
```


DNS message format

Picture from Comer: Internetworking with TCP/IP page 397

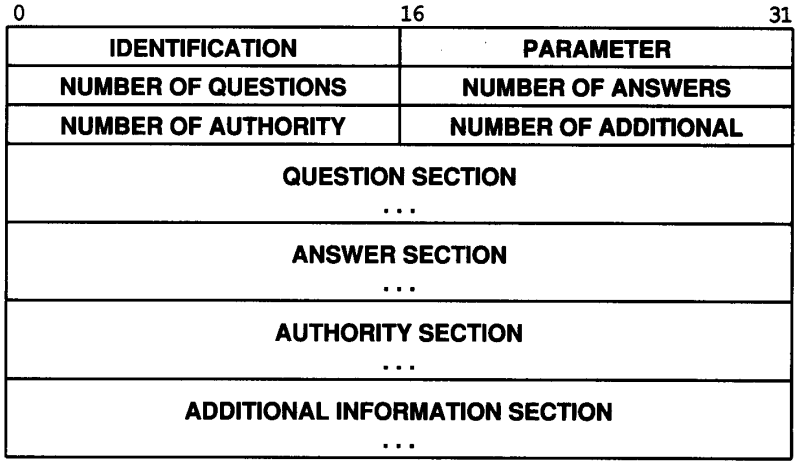


Figure 22.5 Domain name server message format. The question, answer, authority, and additional information sections are variable length.

DNS message format fields

identification	used to match queries to responses
flags	Query or response
	Authorative answer (from authorative server)
	Truncated
	Recursion desired
	Recursion available
	return code (no error or error code)
no. of questions	(must be > 0 for a query)
no. of answers	(must be > 0 for an answer)
question	name being looked up, query type
answer	domain name
	type
	time-to-live
	resource-data-length
	resource data

Representation of a domain name in the DNS protocol

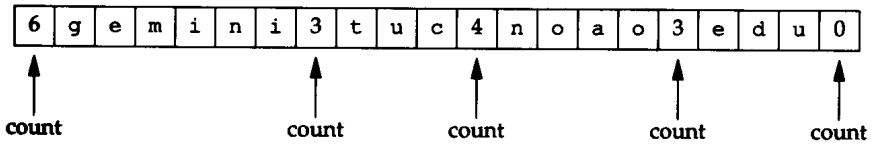


Figure 14.6 Representation of the domain name `gemini.tuc.noao.edu`.

Picture from Stevens: TCP/IP Illustrated, Volume 1, page 193

Weak verification of client's domain

(will only work for clients which have a registered domain address)

If the client has indicated a domain address

Look up this domain address in the DNS

Check that the IP address which the access came from is registered with this domain

If the client has indicated an IP address

Check if the IP address indicated by the client aggrees with the IP address the access came from

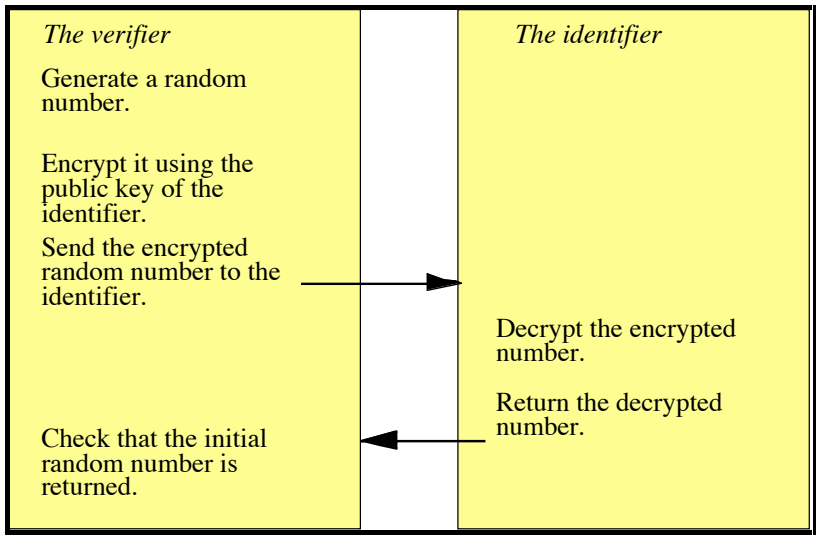
If no match, sometimes you might want to make a reverse DNS lookup to find a domain address for that IP address

Make a non-reverse DNS on the found domain; can return several IP addresses

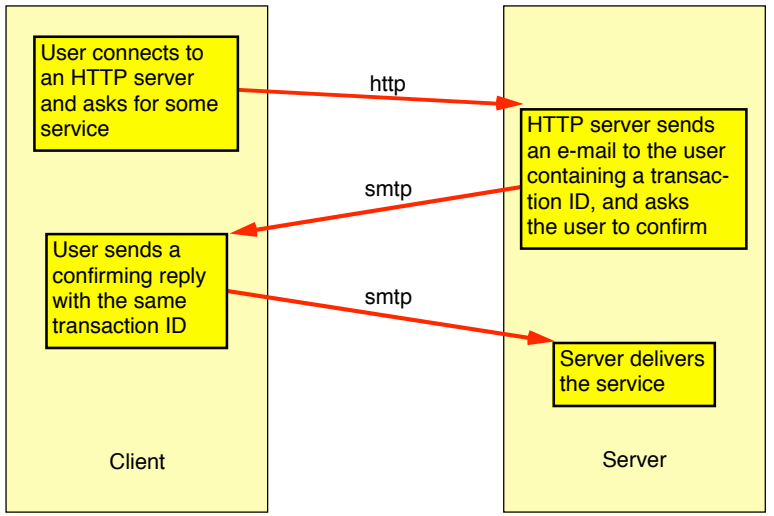
Check that the IP address which the access came from is one of the IP addresses registered with this domain

Basic security services:

Identification (authentication), Authorization, Seals, Signatures, Envelopes (Encryption)



Use of e-mail for authentication



IETF Standards terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Classes of standards

- Experimental standard
- Proposed standard
- Draft standard
- Standard
- Historical
- Informational

BCP (Best Current Practive, cannot go through compatibility testing)

*:96 Overheads

Part 2a: Encoding, ABNF

More about this course about Internet application protocols can be found at URL:

http://dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html

Last update: 04-12-16 12.39

Example of data structure declarations in Pascal


```
flightpointer = ^flight;

flight = RECORD
  airline : String[2];
  flightnumber : Integer;
  nextflight : flightpointer;
END;

passenger = RECORD
  personalname : String [60];
  age : Integer;
  weight : Real;
  gender : Boolean;
 usertexts : ARRAY [1..5] OF
       flightpointer;
END;
```

Why encoding and encoding syntax specification?

1. Syntax must be exact. Saying "Parameters are indicated with a parameter name followed by a parameter value" does not specify whether to encode as "increment 5" or "increment:5" or "increment=5" or "increment = 5".
2. Computer-internal formats like 64-bit floating point are particular to one computer architecture and not portable. Even the storage of octets in 32-bit words is different in different architectures. Sending internal data would thus get "New York" transformed to " weNkroY" when moved between computers with different "byte order".
3. A syntax specification language like ASN.1 ABNF or XML ensures that the syntax specification is unambiguous.

   (Or should be, but ABNF has a historical problem with not fully specifying where white space is allowed, i.e. to distinguish between "From: Peter Paul" and "From:Peter Paul".)

4. Character set must be specified. Defaults are used, but have caused problems.

Character set	Representation of "Ä" (hexadecimal)
ISO Latin One	C4
Unicode (ISO 10646), UCS-4	000000C4
Unicode, UTF-8 coding	E2C4
CP850 (old MS-DOS)	8E
ISO 6937/1	C861
old Mac OS	80

Character sets

A character set is a rule for encoding a certain set of glyphs onto one or more octets. By a glyph is meant a kind of small picture and a kind of syntactic description of the character. The same glyph need not look exactly identical, different fonts can display the same glyph in somewhat different ways.

Examples of characters and their encoding

Syntactic description	Encoding in some common character sets (hexadecimal representation)				Glyphs
	ISO 646	ISO646-SE	ISO 8859-1	Unicode & ISO 10646	
latin capital letter A with diaeresis	n.a.	5B	C4	00C4	Ä
latin capital letter O with diaeresis	n.a.	5C	DC	00DC	Ö
latin capital letter O with stroke	n.a.	n.a.	D8	00D8	Ø
Reverse Solidus	5C	n.a	5C	005C	\

Swedish character encodings

Glyph	å	ä	ö	ü	Å	Ä	Ö	Ü
Two-char encoding	aa	a:	oe	u:	AA	A:	O:	U:
SEN_850200_B = ISO646-SE	7D	7B	7C	??	4D	5B	5C	??
ISO 646 glyph for the encoding above	}	{	|		]	[	\	
ISO 10646	00E5	00E4	00F6	00FC	00C5	00C4	00D6	00DC
ISO 8859-1	E5	E4	F6	FC	C5	C4	D6	DC
Macintosh	8C	8A	9A	9F	81	80	85	86
Old MS-DOS	86	84	94	91	8F	8E	99	9A
T.61=ISO 6937/1	CA61	C861	C86F	C875	CA41	C861	C86F	C855

How can you put more than 255 different characters into eight bit octets?

Method 1	ISO 6937	Use multiple characters for some encodings, for example é as e´ or o as o¨.
Method 2	ISO 2022	Use several different 255 character sets, and special shift sequences to shift from one set to another set.
Method 3	Unicode, ISO 10646	Use two or four octets for each character, but provide compression techniques to compress them during transmission. UTF-8 is an example of a compression encoding scheme for ISO 10646, which has the property that the most common characters like a-z and A-Z, have the same one-octet encoding as in ISO 646 and ISO 8859-1.
Method 4	HTML, MIME Quoted-Printable	Use special encodings for special characters, like ` ` for non-breaking space or `&ouml;` for ö.

UTF-8 encoding of ISO 10646 and Unicode

The UTF-8 (RFC 2044) is an encoding of Unicode with the very important property that all US-ASCII characters have the same coding in UTF-8 as in US-ASCII. This means that protocols, in which special US-ASCII characters have special significance, will work, also with UTF-8. They start with the two or four-octet encodings of ISO 10646 (UCS-4):

UCS-4 range (hex.)	UTF-8 octet sequence (binary)
0000 0000-0000 007F	0xxxxxxx
0000 0080-0000 07FF	110xxxxx 10xxxxxx
0000 0800-0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000-001F FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
0020 0000-03FF FFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0400 0000-7FFF FFFF	1111110x 10xxxxxx ... 10xxxxxx

Subsets used in some standards

Name	Subset description	Where it is used
specials	“(”, “)”, “<”, “>”, “@”, “,”, “;”, “:”, “\”, “"”, “.”, “[”, “]”	Must be coded when used in e-mail addresses.
non-specials	All printable US-ASCII characters except specials and space	Can be used without special coding in e-mail addresses.
Unsafe	"{", "}", "|", "\", "^", "~", "[", "]" and "`"	Must be coded when used in URLs
Reserved	“;”, “/”, “?”, “:”, “@”, “=” and “&”	These characters have special meaning in URLs, and must be coded if used without the reserved meaning.
Safe	All printable US-ASCII characters except Unsafe and Reserved characters and space.	Can be used without special coding in URLs.

Binary and textual data

Binary data

Examples: Data compressed with various compression algorithms, images in formats like GIF, JPEG or TIFF, application data in a format particular to a certain application, such as Word, Excel, Filemaker Pro, Adobe Acrobat, etc.

Textual data

3, 14159 TRUE

Data which is textual in character, in that it consists of a sequence of “readable” characters, sometimes organized into lines, such as plain text, HTML source, Postscript documents, source code in a programming language, etc.

There is no sharp limit between binary and textual data. Some properties which sometimes distinguishes textual data are:

- The character sequence to delimit line breaks differs between platforms, and is often modified at transmission from one platform to another. Macintosh usually uses a single Carriage Return (CR), Unix usually uses a single Line Feed (LF), MS Windows usually uses the character sequence CRLF in file storage, but this is often transformed to only LF when data is important into RAM by an application program.
- Sometimes, characters are encoded according to a character set, which is a rule deciding which glyph to show for a certain bit combination. Sometimes, the character set is modified when textual data is moved between computers or between applications.

Marking the end of data

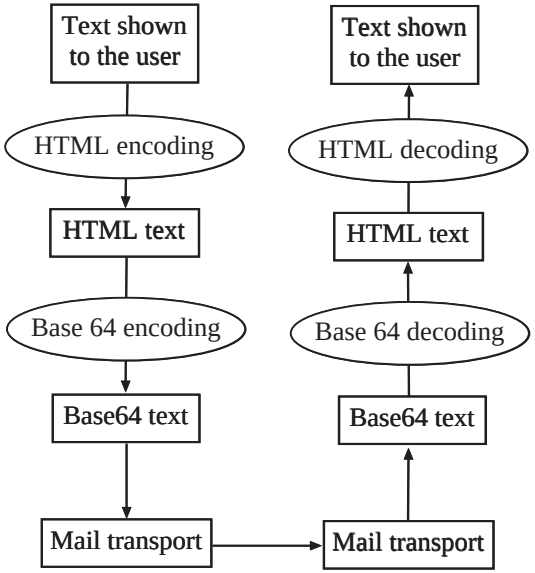
Internet protocols often need to transmit one or several objects of data. The data transmitted is often formatted according to its own encoding rules.

Method	Description	Examples	Used in	Problems
1	Use a special character sequence to mark end of data	`CRLF.CRLF`	SMTP	What to do if this sequence occurs in the data you want to transmit?

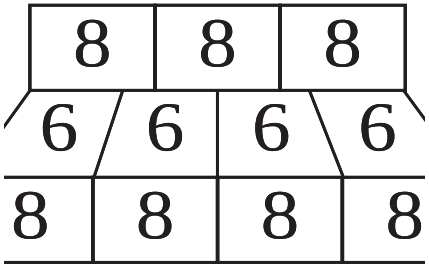
		`boundary: xyzabc`
		`--xyzabc¶`
		`--xyzabc--¶`
			MIME	

2	Indicate length in advance	`10*ABCDEFGHIJ`	HTTP	You may not know the length in advance, for example live broadcasting
3	Chunked transmission	`5*ABCDE5*FGHIJ`	HTTP	
4	Encode in limited character set	`UuRrc232cmflcw`	Base64	Inefficient

Encoding in more than one layer



Base64 encoding of binary data into text

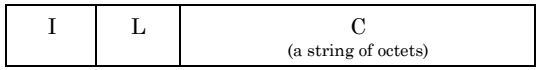


BASE64 is more reliable and works as follows: Take three octets (24 bits), split them into four 6-bit bytes, and encode each 6-bit byte as one character. Since 6-bit bytes can have 64 different values, 64 different characters are needed. These have been chosen to be those 64 ascii characters which are known not to be perverted in transport. Since BASE64 requires 4 octets, 32 bits, to encode 24 bits of binary data, the overhead is 1/3 or 33 %.

Encoding of protocol units

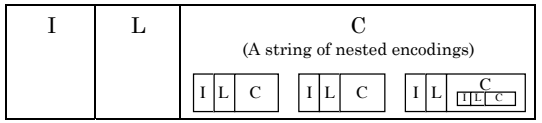
Binary encoding, often in the format: {identifier; length; value}

Primitive:



I = Identifier octets
L = Length octets
C = Contents octets

Constructed:



Binary encodings are often specified using the ASN.1 specification language.

Textual encoding, much like text in a programming language:

Example 1:

```
a002 OK [READ-WRITE] SELECT completed
```

Example 2:

```
EditReplace .Find = "^p ", .Replace = " "
```

Textual encodings are often specified using the ABNF specification language.

Linear White Space

Character name	Real rendering	Notation on this page
Space	A non-printing break with the same width as a single letter.	
Horizontal tab	Moving the printing position to the next print position, usually a wider break than for a space.	→
Line break	Moving the printing position to the next line, using CR, LF or CR+LF.	¶

Acronym	Term	Description	Examples
LWSP	Linear White Space	Sequence of one or more space and horizontal tab characters.	→ →
FWSP	Folding White Space	Linear White Space which also can include line breaks. Continuation lines must begin with tab or space.	→ ¶ →
			¶ →
CFWSP	Comment Folding White Space	Folding White Space which can contain comments in parenthesis.	(Rose) ¶ →(Tulip)

Examples of identical code, in-spite-of CLWSP, in e-mail headers:

```
In-Reply-To: <199807112000.WAA30049@mailbox.hogia.net>
```

```
In-Reply-To: (Your message of 11 July 1998)
             <199807112000.WAA30049@mailbox.hogia.net>
```

```
In-Reply-To: <199807112000.WAA30049@mailbox.hogia.net>
 (Your message of 11 July 1998)
```

Inpreciseness of common usage of where LWSP and CLWSP is allowed and not allowed.

Many different Internet standards use ABNF, but all of them do not use exactly the ABNF notation in the same way. In particular, many Internet standards do not specify where LWSP (Linear White Space) is permitted or required.

Thus, Internet standards often specify things like:

```
Subject = "Subject" ":" "sentence"
```

Is space allowed/required or not between elements here?

The above ABNF specification, when used in older standards, might not clarify if spaces are *allowed* or *required* between the elements.

ABNF syntax elements

A simple ABNF production with an OR ("/") element:

```
answer    = "Answer: " ( "Yes" / "No" )
```

This says that when you send an "answer" from one computer to another, you send either the string `"Answer: Yes"` or the string `"Answer: No"`.

A series of elements of the same kind

There is often a need to specify a series of elements of the same kind. For example, to specify a series of "yes" and "no" we can specify:

```
yes-no-series    = *( "yes " / "no " )
```

This specifies that when we send a yes-no-series from one computer to another, we can send for example one of the following strings (double-quote not included):

```
"yes "                "yes no "
""                    "yes yes yes "
```

The "*" symbol in ABNF means"repeat zero, one or more times"

So yes-no-series, as defined above, will also match an empty string.

A number can be written before the "*" to indicate a minimum, and a number after the "*" to indicate a maximum.

Thus "1*2" means one or two ocurrences of the following construct,
"1*" means one or more, "*5" means between zero and five occurences.

If we want to specify a series of exactly five yes or no, we can thus specify:

```
five-yes-or-no    = 5*5( "yes " / "no " )
```

and if we want to specify a series of between one and five yes or no, we can specify:

```
one-to-five-yes-or-no    = 1*5( "yes " / "no " ) ;Compare *5 1*
```


Linear White SPace (LWSP)

There is often a need to specify that one or more characters which just show up as white space (blanks) on the screen is allowed. In newer standards, this is done by defining Linear White Space:

```
LWSP char    = ( SPACE / HTAB )          ; either one space or one tab
LWSP         = 1*LWSP-char               ; one or more space
                                         characters
```

LWSP, as defined above, is thus one or more SPACE and HTAB characters.

Using LWSP, we can specify for example:

```
yes-no-series    = * (( "yes" / "no" ) LWSP )
```

examples of a string of this format is:

```
"yes "                "yes no "
"no "                 "yes yes yes "
""                    "yes      yes        no    "
```

Comma-separated list

Older ABNF specifications often uses a construct "#" which means the same as "*" but with a comma between the elements. Thus, in older ABNF specifications:

```
yes-no-series     = *( "yes" / "no")
```

is meant to match for example the strings

```
“yes”                “yes no”
“no”                 “yes yes yes”
```

while

```
yes-no-series    = #( "yes" / "no" )
```

is meant to match the strings

```
“yes”                “yes, no”
“no”                 “yes, yes, yes”
```

The problem with this, however, is that neither of the notations above specify where LWSP is allowed. Thus, newer ABNF specifications would instead use:

```
yes-or-no         = ( "yes" / "no" )
yes-no-series     = yes-or-no *( LWSP yes-or-no)
```

to indicate a series of “yes” or “no” *separated by LWSP*, or

```
yes-no-series     = yes-or-no *( "," LWSP yes-or-no)
```

to indicate a series of “yes” or “no” *separated by “,” and LWSP*.

ABNF syntax rules, parentheses

Elements enclosed in parentheses are treated as a single element. Thus, "(elem (foo / bar) elem)" allows the token sequences "elem foo elem" and "elem bar elem".

Example 1 (From RFC2822):

```
keywords        =       "Keywords:" phrase *("," phrase) CRLF
phrase          =       1*word / obs-phrase
word            =       atom / quoted-string
atom            =       [CFWS] 1*atext [CFWS]
```

Example 1, value:

```
Keywords: Orchids, Tropical flowers
```

Example 2 (from RFC822):

```
 authentic   =   "From"    ":"   mailbox     ; Single author
                  / ( "Sender" ":" mailbox ; Actual submittor
                "From" ":" 1#mailbox)      ; Multiple authors
                                           ;  or not sender
```

Example 2, value a:

```
From: Donald Duck <dduck@disney.com>
```

Example 2, value b:

```
Sender: Walt Disney <walt@disney.com>
From:   Donald Duck <dduck@disney.com>
```

Optional elements

There is often the need to specify that something can occur or can be omitted.

This is specified by square brackets. Example:

```
answer    = ("yes" / "no" ) [ ", maybe" ]
```

will match the strings

```
“yes”
“no”
“yes, maybe”
“no, maybe”
```

Square brackets is actually the same as "0*1, the ABNF production above could as well be written as:

```
answer    = ( "yes" / "no" ) 0*1( ", maybe" )
```

or

```
answer    = ( "yes" / "no" ) *1( ", maybe" )
```

Summary of ABNF notation

Notation	Meaning	Example	Meaning
`“/”`	either or	`Yes / No`	Either `Yes` or `No`
`n*m(element)`	Repetition of between n and m elements	`1*2(DIGIT)`	One or two digits
`n*n(element)`	Repetition exactly n times	`2*2(DIGIT)`	Exactly two digits
`n*(element)`	Repetition n or more times	`1*(DIGIT)`	A series of at least one digit
`*n(element)`	Repetition not more than n times	`*4(DIGIT)`	Zero, one, two, three or four digits
`n#m(element)`	Same as n*m but comma-separated	`2#3("A")`	`“A,A”` or `“A,A,A”`
`[ element ]`	Optional emenent, same as `*1(element)`	`[ ";" para ]`	The parameter string can be included or omitted

Exercise 1

Specify, using ABNF, the syntax for a directory path, like

`users/smith/file` or

`users/smith/WWW/file`

with none, one or more directory names, followed by a file name.

SOLUTIONS TO EXERCISES IN COMPENDIUM 6 PAGES 57-66

Exercise 2

Specify, using ABNF, the syntax for Folding Linear White Space, i.e. any sequences of spaces or tabs or newlines, provided there is at least one space or tab after each newline.

Examples:

"→ →"

"→ ¶
→"

" ¶
→"

```
Usage:

From: John Smith <jsmith@foo.bar>

From: John Smith
      <jsmith@foo.bar>
      (typed by Mary Smith)
```

Assume SP = Space, HT = Tab,
CR = Carriage Return, LF = Line Feed

Examples of use of ABNF from RFC 2822

Example 1, ABNF (from RFC 2822):

```
LWSP-char   =  SPACE / HTAB          ; semantics = SPACE
```

Example 2, ABNF (from RFC2822):

```
mailbox          =       name-addr / addr-spec
name-addr        =       [display-name] angle-addr
angle-addr       =       [CFWS] "<" addr-spec ">" [CFWS]
                         / obs-angle-addr
display-name     =       phrase
addr-spec        =       local-part "@" domain
```

Example 2, value a:

```
jpalme@dsv.su.se
```

Example 2; value b:

```
Jacob Palme <jpalme@dsv.su.se>
```


Example 3 (from RFC2822):

```
fields = *(trace
           *(resent-date /
             resent-from /
             resent-sender /
             resent-to /
             resent-cc /
             resent-bcc /
             resent-msg-id))
         *(orig-date /
         from /
         sender /
         reply-to /
         to /
         cc /
         bcc /
         message-id /
         in-reply-to /
         references /
         subject /
         comments /
         keywords /
         optional-field)
```


Example 4 (from RFC2822)

```
in-reply-to      =       "In-Reply-To:" 1*msg-id CRLF
msg-id           =       [CFWS] "<" id-left "@" id-right ">" [CFWS]
id-left          =       dot-atom-text / no-fold-quote / obs-id-left
id-right         =       dot-atom-text / no-fold-literal /
                         obs-id-right
no-fold-quote    =       DQUOTE *(qtext / quoted-pair) DQUOTE
```

Example 4, value a:

```
In-Reply-To: <12345*jpalme@dsv.su.se>
```

Example 4, value b:

```
In-Reply-To: <12345*jpalme@dsv.su.se> <5678*jpalme@dsv.su.se>
```

Example 4, value c:

```
In-Reply-To: Your message of July 26 <12345*jpalme@dsv.su.se>
```

Examples of use of square brackets ([) and (])

Square brackets enclose optional elements; "[foo bar]" is equivalent to "*1(foo bar)".

Example 5 (from RFC822):

```
received    =  "Received"    ":"              ; one per relay
                 ["from" domain]              ; sending host
                 ["by"   domain]              ; receiving host
                 ["via"  atom]                ; physical path
                *("with" atom)                ; link/mail protocol
                 ["id"   msg-id]              ; receiver msg id
                 ["for"  addr-spec]           ; initial form
```

Example 5, value a:

```
Received: from mars.dsv.su.se (root@mars.dsv.su.se
       [130.237.158.10])
       by zaphod.sisu.se (8.6.10/8.6.9) with ESMTP
       id MAA29032 for <cecilia@sisu.se>
```


ABNF syntax rules, comments

A semi-colon, set off some distance to the right of rule text, starts a comment that continues to the end of line.

Example 6 (from RFC2822):

```
specials        =       "(" / ")" /     ; Special characters used in
                        "<" / ">" /     ;  other parts of the syntax
                        "[" / "]" /
                        ":" / ";" /
                        "@" / "\" /
                        "," / "." /
                        DQUOTE
```


Example 7 (from RFC822):

```
obs-zone     =    "UT" / "GMT" /          ; Universal Time
                                          ; North American UT
                                          ; offsets
                  "EST" / "EDT" /         ; Eastern:  - 5/ - 4
                  "CST" / "CDT" /         ; Central:  - 6/ - 5
                  "MST" / "MDT" /         ; Mountain: - 7/ - 6
                  "PST" / "PDT" /         ; Pacific:  - 8/ - 7

                  %d65-73 /               ; Military zones - "A"
                  %d75-90 /               ; through "I" and "K"
                  %d97-105 /              ; through "Z", both
                  %d107-122               ; upper and lower case
```


Exercise 3

Specify the syntax of a new e-mail header field with the following properties:

Name: “Weather”

Values: “Sunny” or “Cloudy” or “Raining” or “Snowing”

Optional parameters: ";" followed by parameter, "=" and integer value

Parameters: “temperature” and “humidity”

Examples:

```
Weather: Sunny ; temperature=20; humidity=50
Weather: Cloudy
```

Exercise 4

An identifier in a programming language is allowed to contain between 1 and 6 letters and digits, the first character must be a letter. Only upper case character are used. Write an ABNF specification for the syntax of such an identifier.

RFC 822 lexical scanner 1


```
CHAR        =  <any ASCII character>          ; (  0-177,  0.-127.)
ALPHA       =  <any ASCII alphabetic character>
                                              ; (101-132, 65.- 90.)
                                              ; (141-172, 97.-122.)
DIGIT       =  <any ASCII decimal digit>      ; ( 60- 71, 48.- 57.)
CTL         =  <any ASCII control             ; (  0- 37,  0.- 31.)
                character and DEL>            ; (    177,    127.)
CR          =  <ASCII CR, carriage return>    ; (     15,     13.)
LF          =  <ASCII LF, linefeed>           ; (     12,     10.)
SPACE       =  <ASCII SP, space>              ; (     40,     32.)
HTAB        =  <ASCII HT, horizontal-tab>     ; (     11,      9.)
<">         =  <ASCII quote mark>             ; (     42,     34.)
CRLF        =  CR LF
```

The same with the 1994 version of ABNF


```
LWSP-char=  SPACE / HTAB                ; semantics = SPACE
ALPHA     = %x41-5A / %x61-7A ; A-Z / a-z
BIT       = "0" / "1"
CHAR      = %x01-7F    ; any 7-bit US-ASCII character, excluding NUL
CR        = %x0D             ; carriage return
CRLF      = CR LF            ; Internet standard newline
CTL       = %x00-1F / %x7F; controls
DIGIT     = %x30-39          ; 0-9
DQUOTE    = %x22             ; " (Double Quote)
HEXDIG    = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
HTAB      = %x09             ; horizontal tab
LF        = %x0A             ; linefeed
LWSP      = *(WSP / CRLF WSP) ; linear white space (past newline)
OCTET     = %x00-FF          ; 8 bits of data
SP        = %x20
```


%d13 %x0D	he
	carriage return character.
b110 1	is the character with binary value 1101, which is a third way of specifying the carriage return character.
%x30- 39	means all characters with hexadecimal values from 30 to 39, which is the digits 0-9 in the ASCII character set.
%d13.10	is a short form for %d13 %d10, which is carriage return followed by line feed.

*:96 Overheads

Part 2ca: Extensible Markup Language (XML)

More about this course about Internet application protocols can be found at URL:

http://dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html

Last update: 00-03-24 17.24

HTML Example

```
<h2>False Pretences</h2>
<p><b>By: </b>Margaret Yorke<br>
<b>ISBN: </b>0-312-19975-9<br>
<b>Year: </b>1999</p>
```

XML Example

```
<book><author><surname>Yorke</surname>
<given-name>Margaret</given-name></author>
<title>False Pretences</title>
<isbn>0-312-19975-9</isbn>
<year>1999</year></book>
```

The difference between HTML and XML: In XML you can yourself decide which tags to use. In HTML, you can only use the built-in tags specified in HTML. In the example above, I used the tags `<book>`, `<author>`, `<surname>`, `<given-name>`, `<title>`, `<isbn>` and `<year>`. In another application, I could have chosen other tags.

By `combining` of XML with style sheets, you can still get the documented printed in the same way as if you had been using HTML.

Uses of XML

- For transport of information between data bases.
- For sending of information to be displayed to a user, just like with HTML.
- As a rather readable format in itself (except for encoding of special characters).
- For encoding of network operations, as an alternative to ABNF or ASN.1.

Restrictions of XML

- Binary data must be either encoded as BASE64 or sent outside of the XML document (like in HTML).
- A rather wordy format, but compression can reduce this.

Some acronyms

Standard Generalized Markup Language (SGML)

HTML and XML are both simplifications of SGML.

Application Program Interfaces (APIs)

Document Object Model (DOM) is an API för XML. Supported by newer web browsers.

Simple API for XML (SAX) standard for API to XML partsers. Streambased - the XML content is delivered in increments during its interpretation.

Style sheet languages

The same XML document can be shown in different formats, by using different style sheets.

eXtensible Style Sheet Language (XSL).

eXtensible Style Sheet Language Transformatsions (XSLT).

Cascading Style Sheet, level 1 och 2 (CSS1, CSS2).

Function	HTML	XML
Set of tags	Built-in, predefined set.	Every application can define its own element types and select their tags.
End tag	Not always required.	Always required.
Case sensitive	No, for example, <TITLE> and <title> are identical.	Yes, <TITLE> and <title> are different and <TITLE> must end with </TITLE>, not with </title>.
Coding errors	Often accepted	Not accepted
Support in web browsers	Yes.	Yes in some newer versions.
Text layout and style	HTML tags and style sheets.	Style sheets and XSLT transformation code.

Basics of the XML format

XML facility:	Example:
User-selected tags.	<book>, <songs>, <position> or whatever you need for your data.
Tags can have attributes.	<book author="Margaret Yorke" title="False Pretences">
Tags which have no embedded data can be closed in the opening tag.	<book author="Margaret Yorke" title="False Pretences"/> instead of <book author="Margaret Yorke" title="False Pretences"></book>
Tags can be nested.	<book><author>Margaret Yorke</author>...</book>
Tags must be closed.	Not correct: <book><author>Margaret Yorke</book>
Certain special character must be encoded.	<book title="The "queen"of Sheba"/>

Exercise 41

Here is an example of part of an e-mail heading according to current e-mail standards.

```
From: Nancy Nice <nnice@good.net>
To:   Percy Devil <pdevil@hell.net>
Cc:   Mary Clever <mclever@intelligence.net>,
      Rupert Happy <rhappy@fun.net>
```

How might the same information be encoded using XML?

XML is more strict than accepted HTML practice

HTML browsers accept many kinds of formally illegal HTML encodings. This is not allowed in XML. Examples:

```
Legal: <p>First paragraph.</p><p>Second paragraph</p>
Accepted: <p>First paragraph.<p>Second paragraph</p>

Legal: <b><i>Bold and Italics</i></b>
Accepted: <b><i>Bold and Italics</b></i>

Legal: <FONT COLOR="#FFFF66">
Accepted: <FONT COLOR=#FFFF66>
```

Tags are case-sensitive in XML

```
Illegal: <H1>Heading text</h1>
Legal: <H1>Heading text</H1>
```

White space is sometimes relevant in #PCDATA, but normalized in attributes

```
<CHRISTMAS>
     X
    XXX
   XXXXX
```

```
<CHRISTMAS FATHER="Donald
Duck">
```

is identical to

```
<CHRISTMAS FATHER="Donald Duck">
```

Document Type Definition (DTD)

An XML document may be connected with a document type definition. But this is not mandatory, you can send XML data without a DTD.

The DTD describes the allowed syntax, i.e. the tags and their allowed attributes.

Example of a DTD

```
<!ELEMENT book (author+)>
<!ATTLIST book
  title CDATA #REQUIRED
  year CDATA #IMPLIED >
<!ELEMENT author (#PCDATA)>
```

Example of XML using this DTD

```
<?xml version="1.0" ?>
<!DOCTYPE book SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/book.dtd">
<book title="False Pretences" year="1999" >
<author>Margaret York</author>
</book>
```



```
From: Nancy Nice <nnice@good.net>
To:   Percy Devil <pdevil@hell.net>
Cc:   Mary Clever <mclever@intelligence.net>,
      Rupert Happy <rhappy@fun.net>
```

Exercise 41 solution

```
<?xml version="1.0" ?>
<!DOCTYPE header SYSTEM "header.dtd">
<header>
  <from>
     <person>
        <user-friendly-name>Nancy Nice</user-friendly-name>
        <local-id>nnice</local-id><domain>good.net</domain>
     </person></from>
  <to>
     <person>
        <user-friendly-name>Percy Devil</user-friendly-name>
        <local-id>pdevil</local-id><domain>hell.net</domain>
     </person></to>
  <cc>
     <person>
        <user-friendly-name>Mary Clever</user-friendly-name>
        <local-id>mclever</local-id>
        <domain>intelligence.net</domain>
     </person><person>
        <user-friendly-name>rupert happy</user-friendly-name>
        <local-id>rhappy</local-id><domain>fun.net</domain>
     </person></cc>
</header>
```

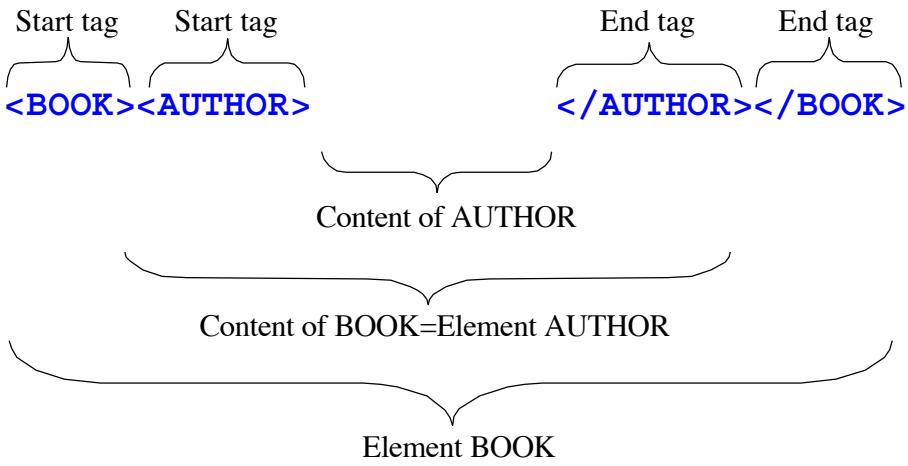

Relation between DTD and XML

Enviroment:	"ABNF"	"ASN.1"	"XML"
Language for specifying the encodings for a particular application.	ABNF	ASN.1	DTD (but not mandatory, and not as strong typing as in ASN.1)
Language used to actually encode data.	Text, often as a list of lines beginning with a name, a colon, followed by a value.	BER (or some other ASN.1 encoding rule)	XML

Special Character Encoding in XML

Reserved character	Predefined entity to use instead
<	<
&	&
>	>
'	'
"	"

ELEMENT and TAG



How to specify the DTD in the XML using it

```
<?xml version="1.0"?>
```

Specifies that this is XML-encoded data

```
<!DOCTYPE person
SYSTEM "person.dtd">
```

Specifies where to find the DTD. "Person.dtd" can be a complete URL, which gives a globally unique reference to this DTD.

```
<PERSON>
  <NAME>John Smith</NAME>
  <BIRTHYEAR>1941</BIRTHYEAR>
  <WAGE>57000</WAGE>
</PERSON>
```

Here comes the XML encoded according to this DTD.

DTD ELEMENT with subelements

`(a,b)` means the element `a` followed by the element `b`.

Example of a DTD

```
<!ELEMENT author (givenname,surname)>
<!ELEMENT givenname (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE author SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/author.dtd">
<author>
<givenname>Margaret</givenname>
<surname>York</surname>
</author>
```

Well-formed = correct XML, but need not have any DTD

Valid = correct XML and in accordance with a specified DTD

DTD ELEMENT with free text content

Example of a DTD

```
<!ELEMENT author (#PCDATA)>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE author SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/author.dtd">
<author>Margaret York</author>
```

Example 2 of XML using this DTD

```
<author>Text containing &lt; special markup &gt;</author>
```

Example 3 of XML using this DTD

```
<author>
<![CDATA[
Text containing < special markup > like & and " and '
]]>
</author>
```

XML Fortsätter på sid 172

DTD ELEMENT with subelements

(a?) means that the element a is repeated 0 or 1 times.

Example of a DTD

```
<!ELEMENT basic-family (father?,mother?,child*)>
<!ELEMENT father (#PCDATA)>
<!ELEMENT mother (#PCDATA)>
<!ELEMENT child (#PCDATA)>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE basic-family SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/basic-
family.dtd">
<basic-family>
<father>John</father>
<child>Eve</child>
<child>Peter</child>
</basic-family>
```


Exercise 42

Write a DTD for an XML-variant of the e-mail header in Exercise 41.

```
From: Nancy Nice <nnice@good.net>
To: Percy Devil <pdevil@hell.net>
Cc: Mary Clever
    <mclever@intelligence.net>,
    Rupert Happy <rhappy@fun.net>
```


DTD ELEMENT with subelements

(a*) means that a is repeated 0, 1 or more times.

Example of a DTD

```
<!ELEMENT family (father,mother,child*)>
<!ELEMENT father (#PCDATA)>
<!ELEMENT mother (#PCDATA)>
<!ELEMENT child (#PCDATA)>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE family SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/family.dtd">
<family>
<father>John</father>
<mother>Margaret</mother>
<child>Eve</child>
<child>Peter</child>
</family>
```


DTD ELEMENT with subelements

(a+) means that a is repeated 1 or more times.

Example of a DTD

```
<!ELEMENT child-family (father,mother,child+)>
<!ELEMENT father (#PCDATA)>
<!ELEMENT mother (#PCDATA)>
<!ELEMENT child (#PCDATA)>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE child-family SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/child-
family.dtd">
<child-family>
<father>John</father>
<mother>Margaret</mother>
<child>Eve</child>
<child>Peter</child>
</child-family>
```

Exercise 43

Specify DTD and an XML example for a protocol to send either a name (single string), a social-security number (another single string) or both.

Exercise 43 solution page A

DTD specification:	XML examples:
<!ELEMENT id (name | social-security-no | both)> <!ELEMENT both (name, social-security-no)> <!ELEMENT name (#PCDATA)> <!ELEMENT social-security-no (#PCDATA)>	**<?xml version="1.0" ?> <!DOCTYPE id SYSTEM "id.dtd"> <id><social-security-no>410201-1410</social-security-no></id>**
	<?xml version="1.0" ?> <!DOCTYPE id SYSTEM "id.dtd"> <id><both><name>Eliza Doolittle</name> <social-security-no>410201-1410</social-security-no></both></id>
	<?xml version="1.0" ?> <!DOCTYPE id SYSTEM "id.dtd"> <id><name>Eliza Doolittle</name> </id>

Exercise 43

Specify DTD and an XML example for a protocol to send either a name (single string), a social-security number (another single string) or both.

Exercise 42 solution

```
<!ELEMENT header (from, to?, cc?)>
<!ELEMENT from (person)>
<!ELEMENT to (person+)>
<!ELEMENT cc (person+)>
<!ELEMENT person (user-friendly-name,local-id,domain)>
<!ELEMENT user-friendly-name (#PCDATA)>
<!ELEMENT local-id (#PCDATA)>
<!ELEMENT domain (#PCDATA)>
```

Exercise 42

Write a DTD for an XML-variant of the e-mail header in Exercise 41.

```
From: Nancy Nice <nnice@good.net>
To: Percy Devil <pdevil@hell.net>
Cc: Mary Clever
    <mclever@intelligence.net>,
    Rupert Happy <rhappy@fun.net>
```


DTD ELEMENT with subelements

"|" means either-or "," means succession.

EMPTY (without parenthesis) means no contained data.

Example of a DTD

```
<!ELEMENT operations (((get | put),uri)*)>
<!ELEMENT get EMPTY>
<!ELEMENT put EMPTY>
<!ELEMENT uri (#PCDATA)>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE operations SYSTEM
"http://dsv.su.se/jpalme/internet-course/xml/operations.dtd">
<operations>
<get/><uri>http://cmc.dsv.su.se/file1</uri>
<get/><uri>http://cmc.dsv.su.se/file2</uri>
<put/><uri>http://cmc.dsv.su.se/file3</uri>
</operations>
```

Note: `<get/>` *is a short form for* `<get></get>`

Any Specification

The ANY specification (example:

<!ELEMENT miscellaneous ANY>)

allows any kind of un-specified XML content. This specification should in most cases be avoided, since it makes it difficult for software to check or interpret the content.

Exercise 43 solution page B

Note: The following will not work:

```
<!ELEMENT id ( name |
social-security-no |
(name, social-security-
no))>
<!ELEMENT name (#PCDATA)>
<!ELEMENT social-
security-no (#PCDATA)>
```

This will not work, because the receiving program will not be able to know, when it starts to scan <name> whether this is the first or the third branch of the choice.

DTD ELEMENT with XML attributes

Example of a DTD

```
<!ELEMENT book EMPTY>
<!ATTLIST book
  title CDATA #REQUIRED
  author CDATA 'anonymous'
  weight CDATA #IMPLIED
  format (paper-back | hard-back) 'paper-back'
>
```

Example 1 of XML using this DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE book SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/book.dtd">
<book
  title="False Pretences"
  author="Margaret Yorke"
  format="hard-back"
/>
```


Operators in lists of subelements:

Code:	Explanation:
a, b	Mandatory a followed by mandatory b.
a | b	Either a or b.
a*	0, 1 or more occurences of a.
a+	1 or more occurences of a.
a?	0 or one occurences of a.

Type:	Example:	Description:
IDREFS	`<!ATTLIST author authorid ID #REQUIRED>` `<!ATTLIST book authorids IDREFS #REQUIRED>`	Similar to `IDREF`, but allows a list of more than one value. Needed in this example, if a book can have more than one author.

ENTITY	DTD text:
	`<!ELEMENT LOGO EMPTY>`
	`<!ATTLIST LOGO GIF-FILE ENTITY #REQUIRED>`
	`<!ENTITY DSV-LOGO SYSTEM "dsv-logo.gif">`
	XML text:
	`<LOGO GIF-FILE="DSV-LOGO"/>`
		This is one way to include binary data in an XML file, by referring to the URI of the binary data. Just like with `<IMG>` tags in HTML, the actual binary file is not included, just referenced.

Type:	Example:	Description:

ENTITIES	DTD text:
	`<!ELEMENT LOGO EMPTY>`
	`<!ATTLIST LOGO GIF-FILE ENTITIES #REQUIRED>`
	`<!ENTITY DSV-LOGO SYSTEM "dsv-logo.gif">`
	`<!ENTITY KTH-LOGO SYSTEM "kth-logo.gif">`
	XML text:
	`<LOGO GIF-FILE="DSV-LOGO KTH-LOGO"/>`
		A list of more than one entity.

NMTOKEN	`<!ATTLIST variable-name #NMTOKEN>`	A name, formatted like a variable name in a computer program. Useful when you use XML to generate source program code.
NMTOKENS	`<!ATTLIST variables #NMTOKENS>`	A list of names, similar as for NMTOKEN above.
NOTATION	`<!ATTLIST SPEECH PLAYER NOTATION ( MP3 | QUICKTIME ) #REQUIRED>`	The name of a non-XML encoding.

Default values for XML attributes

DTD term:	Example:	Description:
A single value within quotes at the end of the attribute.	`<!ATTLIST book binding (hardback | paperback) "hardback">`	This default value should be assumed if the attribute is not specified in the XML text.
#REQUIRED	`<!ATTLIST book binding (hardback | paperback) #REQUIRED>`	No default value is allowed, the attribute must always be specified in the XML text.
#IMPLIED	`<!ATTLIST book binding (hardback | paperback) #IMPLIED>`	No default value, but the attribute is not required. If the attribute is not given, this might mean that it is unknown or not valid.
#FIXED	`<!ATTLIST book binding (hardback | paperback) #FIXED "hardback">`	The XML can either contain this attribute or not, but if it is there, it must always have this particular value.

Types of XML attributes

Type:	Example:	Description:
CDATA	`<!ATTLIST book title #REQUIRED>`	Any character string.
A list of enumerated values	`<!ATTLIST book binding (hardback | paperback) "hardback">`	Restricted to the listed values only.
ID	`<!ATTLIST book entryno ID #REQUIRED>`	Gives a name to this particular element. No other element in the XML text can have the same name. Unique names on elements are useful in some cases for programs which manipulate the XML text.
IDREF	`<!ATTLIST author authorid ID #REQUIRED>` `<!ATTLIST book authorid IDREF #REQUIRED>`	Reference to the unique name, which was given to another element in the XML text. In the example, every element of type author has an `ID` authorid, and every element of type book has an `IDREF` referring to the `ID` of the element for the author of that book.

Exercise 44 solution

DTD specification:

```
<!ELEMENT movie (title,
  person+)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT person EMPTY>
<!ATTLIST person
  name CDATA #REQUIRED
  role (actor | photographer |
  director | author |
  administrator)
  #IMPLIED
>
```

XML data:

```
<?xml version="1.0" ?>
<!DOCTYPE movie SYSTEM
  "movie.dtd">
<movie>
<title>
The Postman Always Rings
Twice</title>
<person name="Lana Turner"
   role="actor"/>
<person name="John Garfield"
   role="actor"/>
<person name="Tay Garnet"
   role="director"/>
<person name="James M. Cain"
   role="author"/>
</movie>
```

Exercise 44

Specify DTD and an XML example for a protocol to send a record describing a movie. The record contains a title and a list of people. Each person is identified by the attributes name, and optionally, the attribute role as either actor, photographer, director, author or administrator. As an XML example, use the movie "The Postman Always Rings Twice", directed by Tay Garnet based on a book by James M. Cain with leading actors Lana Turner and John Garfield.

ENTITIES

Built-in character entities

Example: `&quot; &`

Internal entities

You can add your own additional entity declarations to represent characters or sequences of characters. For example:

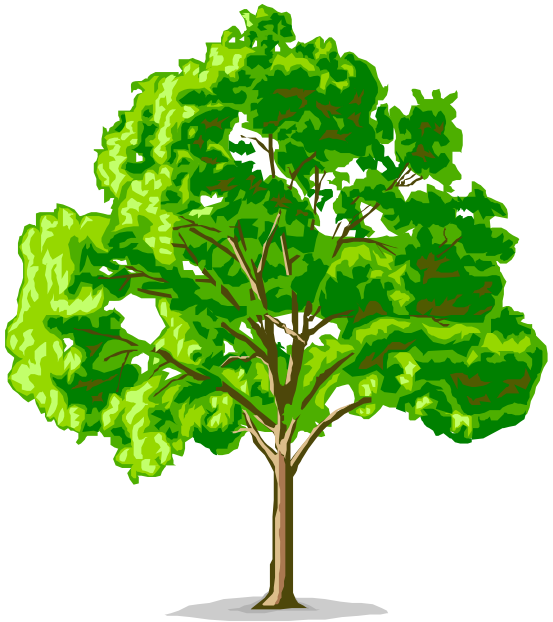
```
<!ENTITY KTH "Kungliga Tekniska Högskolan">
<DESCRIPTION>&KTH; is a technical university.</DESCRIPTION>
```

is identical to

```
<DESCRIPTION>Kungliga Tekniska Högskolan is a technical
university.</DESCRIPTION>
```

External entities

```
<!ENTITY polisvåld SYSTEM
"http://www.palme.nu/free/pv.html">

<!ENTITY comic SYSTEM
"http://www.palme.nu/comics/a-11.gif" NDATA GIF87A>
```

Elements versus attributes

```
<book><author><surname>
Yorke</surname><given-
name>Margaret</given-
name></author></book>
```

versus

```
<book author="Margaret
Yorke"/>
```

Elements are like a tree with branches, each branch can split into new branches.

Attributes are like leaves or fruits, they are the end point, cannot be split further. They also give some rudimentary type control.

Exercise 44

Specify DTD and an XML example for a protocol to send a record describing a movie. The record contains a title and a list of people. Each person is identified by the attributes name, and optionally, the attribute role as either actor, photographer, director, author or administrator. As an XML example, use the movie "The Postman Always Rings Twice", directed by Tay Garnet based on a book by James M. Cain with leading actors Lana Turner and John Garfield.

Use of entities to reference external DTD files

Example of the DTD book.dtd

```
<!ELEMENT book EMPTY>
<!ATTLIST book
  title CDATA #REQUIRED author CDATA 'anonymous'
  weight CDATA #IMPLIED
  format (paper-back | hard-back) 'paper-back' >
```

Example of the DTD collection.dtd

```
<!ENTITY % book SYSTEM "book.dtd">
 %book;
<!ELEMENT collection (book+)>
<!ATTLIST collection owner CDATA #REQUIRED >
```

Example of XML using these DTDs

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE collection SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/collection.d
td">
<collection
  owner="Kungliga Biblioteket"
```



```
>
<book
  title="False Pretences"
  author="Margaret Yorke"
  format="hard-back"
/>
<book
  title="Act of Violence"
  author="Margaret Yorke"
  format="paper-back"
/>
</collection>
```


IDs in XML

Unique names can be used to refer between different places in a document.

XML example:

```
<author ref="myorke">Margaret Yorke</author>
...
<book author="myorke">False Pretences</book>
```

Based on the DTD:

```
<!ELEMENT author (#PCDATA)>
<!ATTLIST author
  ref ID #REQUIRED>
<!ELEMENT book (#PCDATA)>
<!ATTLIST book
  author IDREF #IMPLIED>
```

Attribute types:

ID = Name of this object

IDREF = One single ID reference

IDREFS = List of names separated by white space

NMTOKEN, NMTOKENS = Single words or lists of words separated by white space

More information about XML

The official XML standards specification (rather difficult to read):

http://www.w3.org/TR/REC-xml

Norman Walsh's XML tutorial:

http://www.xml.com/xml/pub/98/10/guide1.html

Rolf Pfeiffer's XML tutorial:

http://www.software.ibm.com/developer/education/tutorial-prog/abstract.html

Doug Tidwell's XML tutorial:

http://www.software.ibm.com/developer/education/xmlintro/

Validator of DTD/XML encodings:

http://www.stg.brown.edu/service/xmlvalid/

*:96 Overheads

Part 2b: Encoding using ASN.1

More about this course about Internet application protocols can be found at URL:

http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html

Last update: 01-12-20 13.57

Till studerande från KTH:

Du måste göra ”Kursval” hos EIT-s kansli, för att kunna få betyg för deltagande i den här kursen.

Compendium 8 page 34C

ASN.1-s historia

(1) Courier, protokoll inom Xerox Corp.

(2) CCITT Recommendation X.409 1984

(3) ISO delar upp X.409 i två standarder, en (ISO 8824) för språket och en (ISO 8825) för BER.

(4) CCITT ger ut dessa 1988 som X.208 och X.209.

(5) Ny version 1994, innehåller bl.a. ny notation som ersättning för macros.

Några Internet standarder som använder ASN.1

Kerberos — Security system (Authentication, etc.) [RFC 1510]

LDAP — Lightweight Directory Access Protocol [RFC 1777]

SNMP — Simple Network Management Protocol [RFC 1303]

SMIME — Security Enhanced MIME

ASN.1 versus ABNF

Similarities

1. Both are languages for the specification of the syntax of encoded data transmitted between computers in net-based protocols.
2. Both are based on the BNF (Backus-Naur-Form) form syntax specifications, which was first used in the Algol-60 specification.

Differences

1. ABNF specifies encoding of information into text strings, ASN.1 specified encodings of information into usually binary form (octet strings). Because of this difference, ABNF encodings are easier for a human to read.
2. ASN.1 specifies a tag-length-value kind of encoding, which avoids many problems with delimiters and delimeter encoding in ABNF.

Nya datatyper definieras ur kända typer


```
Temperature ::=     REAL - - in degrees Kelvin
```

One-component data types

```
Temperature ::=     [APPLICATION 0] REAL - - in degrees Kelvin
WindVelocity ::=    [APPLICATION 1] REAL - - in m/s
Humidity ::=        [APPLICATION 2] REAL - - relative percentage
```

Two-component data types

```
ComplexNumber::=[APPLICATION 3] SEQUENCE
                {   imaginaryPart       REAL,
                    realPart            REAL }
```

Använda tidigare definierade typer i nya typdefinitioner

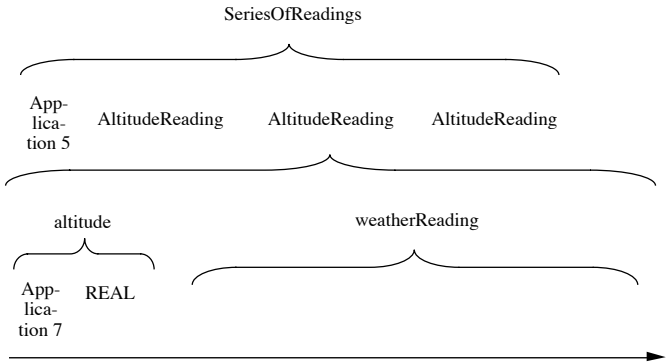

```
Temperature         ::= [APPLICATION 0] REAL - - in degrees Kelvin
WindVelocity        ::= [APPLICATION 1] REAL - - in m/s
Humidity            ::= [APPLICATION 2] REAL - - relative percentage

WeatherReading      ::= [APPLICATION 4] SEQUENCE
                    {   temperatureReading  Temperature,
                        velocityReading     WindVelocity,
                        humidityReading     Humidity }
```

Stor och liten bokstav är signifikant. Första bokstaven måste vara stor för datatyp, liten för datafält (se ovan).

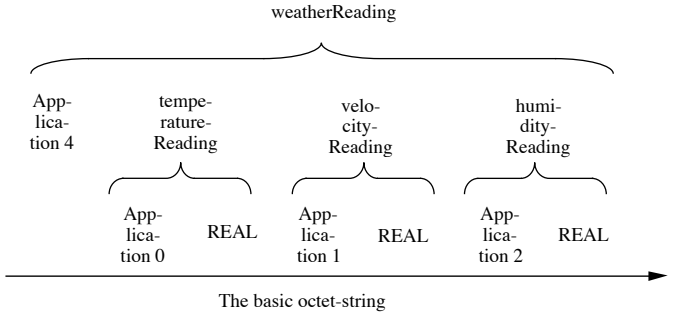
En sekvens av element av samma typ


```
Altitude            ::= [APPLICATION 7] REAL - - Meters
                                        - - above the sea

SeriesOfReadings    ::= [APPLICATION 5] SEQUENCE OF
                        AltitudeReading

AltitudeReading     ::= [APPLICATION 6] SEQUENCE
                    {   altitude        Altitude,
                        weatherReading  WeatherReading }
```

Uppdelning av oktett(bit)-strömmen



Vidare uppdelning av oktettströmmen



Värderepresentation

Vi har tidigare visat exempel på definitioner av datatyper och datafält i ASN.1. Även datavärden kan definieras.

```
Fingers          ::= [APPLICATION 8] INTEGER  - -  hands+feet
humanFingers     Fingers ::= 20
pi               REAL ::= {
                          3141592653589793, 10, -16}
```

Värderepresentation används bl.a. för:

- Förval (defaults)
- Exempel
- Vissa speciella fall, t.ex. objekt-identifierare, gränsvärden, subtypsdefinitioner

Varför används typer oftare än värden i ASN.1-texter?

Ett värde av typen AltitudeReading kan t.ex. vara:


```
{
altitude          {100, 10, 0}
weatherReading    {
                  temperatureReading
                                    {2731,10,-1}
                  velocityReading   {5, 10, 0}
                  humidityReading   {40,10,-2}
                  }
}
```

Terminologi

En *typ* eller *datatyp* är en mängd av värden.

En typ kan definieras genom att räkna upp alla tillåtna värden, eller definieras att ha obegränsat antal värden som t.ex. typerna *Integer* och *Real*.

En ny typ, som definieras som en kombination av element av tidigare definierade typer, kallas för en *strukturerad typ*.

Exempel på strukturerad typdefinition:

```
ComplexNumber::=    [APPLICATION 3] SEQUENCE
                 {  imaginaryPart    REAL,
                    realPart         REAL }
```


Abstract och Transfer syntax

Dokumentformat	Språk	Användning
Notation	Abstract Syntax (ASN.1)	Specifikationer
Kodning	Transfer Syntax (ASN.1-BER)	Kommunikation

ASN.1 = Abstract Syntax Notation 1
BER = Basic Encoding Rules

ASN.1-produktioner

En ASN.1-produktion är en regel, som definierar en typ ur andra typer. Syntaxen för en ASN.1-produktion är

(1) Namnet på den nya typen (börjar med stor bokstav)
(2) Operatorn ”::=”
(3) Definitionen av den nya typen (fältnamn börjar med små bokstäver)

Exempel:

```
ComplexNumber::=    [APPLICATION 3] SEQUENCE
                 {  imaginaryPart    REAL,
                    realPart         REAL }
                    realPart         REAL }
```


Background to Exercise 1-3

```
Temperature  ::= [APPLICATION 0] REAL -- in ° Kelvin
WindVelocity ::= [APPLICATION 1] REAL -- in m/s
Humidity     ::= [APPLICATION 2] REAL -- relative percent
Altitute     ::= [APPLICATION 3] INTEGER -- in meters

WeatherReading   ::= [APPLICATION 4] SEQUENCE
                 {   temperatureReading  Temperature,
                     velocityReading     WindVelocity,
                     humidityReading     Humidity,
                     altitude            Altitude }
```

Exercise 1

You are to define a protocol for communication between an automatic scale and a packing machine.

The scale measures the weight in grams as a floating point number and the code number of the merchandise as an integer.

Define a data type **ScaleReading** which the scale can use to report this to the packing machine.

Exercise 2

Some countries use, as an alternative to the metric system, a measurement system based on inches, feet and yeards. Define a data type **Measurement** which gives one value in this system, and **Box** which gives the height, length and width of an object in this measurement system. Feet and yards are integers, inches is a decimal value (=floating point value with the base 10).

Moduler

<modulnamn> DEFINITIONS ::= BEGIN
<modulkropp>
END

Exempel:

EmptyModule DEFINITIONS ::= BEGIN
END

Typer i ASN.1:

Enkla typer	**Teckensträngs typer**	**Strukturerade typer**	**”Useful types”**
BOOLEAN	NumericString	SET	GeneralizedTime
INTEGER	PrintableString	SET OF	UTCTime
ENUMERATED	TeletexString	SEQUENCE	EXTERNAL
REAL	VideotexString	SEQUENCE OF	ObjectDescriptor
BIT STRING	VisibleString	CHOICE	
OCTET STRING	IA5String	ANY	*Warning: Constraints are strongly recommended for Graphic, General, Universal, BMP and UTF8 strings*
NULL	GraphicString	[Tagged]	
OBJECT IDENTIFIER	GeneralString	*<Different variants < of ISO 10646, not < in the 1998 < version*	
	UniversalString		
	BMPString		
	UTF8String		
	CharacterString		

Reserverade ord

BOOLEAN	**INTEGER**	**BIT**	**STRING**
OCTET	**NULL**	**SEQUENCE**	**OF**
SET	**IMPLICIT**	**CHOICE**	**ANY**
EXTERNAL	**OBJECT**	**IDENTIFIER**	**OPTIONAL**
DEFAULT	**COMPONENTS**	**UNIVERSAL**	**APPLICATION**
PRIVATE	**TRUE**	**FALSE**	**BEGIN**
END	**DEFINITIONS**	**EXPLICIT**	**ENUMERATED**
EXPORTS	**IMPORTS**	**REAL**	**INCLUDES**
MIN	**MAX**	**SIZE**	**FROM**
WITH	**COMPONENT**	**PRESENT**	**ABSENT**
DEFINED	**BY**	**PLUS-INFINITY**	
MINUS-INFINITY		**TAGS**	

Stora resp. små bokstäver är signifikanta, så ett enkelt sätt att undvika kollission med reserverade ord är att använda identifierare som innehåller små bokstäver helt eller delvis.

Identifierarformat

Teckenmängd	Typdefinition	Fält, värde
"a"-"z"	Kan ingå	Kan ingå, måste börjar med
"A"-"Z"	Kan ingå, måste börja med	Kan ingå
"0"-"9"	Kan ingå	Kan ingå
"-"	Kan ingå, men aldrig två i följd	Kan ingå, men aldrig två i följd

Kommentarer

Inleds med "- - ", avslutas med "- - " eller vid radens slut

Integer — Simple Type

Värdemängd: Alla positiva och negativa heltal inklusive 0.
OBS: Ingen maximigräns!

Typnotation:	**Värdenotation:**
INTEGER	**<num>**
INTEGER { <ident> (<num>), ...}	**<ident>**
Exempel:	**Exempel:**
INTEGER **{ touristClass (-1),** **businessClass(0),** **firstClass(1) }**	**4711** **-4294967296** **0** **firstClass**

Subtyper: **Single value, Contained subtype, Range**

Subtyper

Notation:

<type> <subtype-spec>

där <**subtype-spec>** har formen **(<value-set> | ...)**

där <**value-set>** kan vara

- single value
- contained subtype

Bara för vissa typer även

- value range
- size range
- permitted alphabet
- inner subtyping

Subtyper till Integer I

Enkelt värde:

StandardBase ::= INTEGER (2 | 10)

Inkluderad subtype

ExtendedBase ::= INTEGER (INCLUDES StandardBase | 8 | 16)

Value Range (bara när ordning är definierad)

Positive ::= INTEGER (1 .. MAX)
Non-negative ::= INTEGER (0 .. MAX)
Number ::= INTEGER (0 .. <1000)

MAX och **MIN** betyder obegränsat (inte samma sak som +∞ och -∞, kan ej användas som värde, utan bara i Range-satser)!

Subtyper till Integer II

Value Range med namngivna delfält

DayOfTheMonth ::= INTEGER { first(1), last(31) } (first .. last)

Användande av definierade konstanter

```
CharacterPosition ::= INTEGER { first(1), last(lineLength) }
                      (first .. last)

lineLength INTEGER ::= 80
```

Examples of use of subtyping of Integer

Month-number ::= INTEGER (1 .. 12)
months-of-the-year ::= 12 **Month-number ::= INTEGER (1 .. months-of-the-year)**
Single-digit-prime ::= INTEGER (2 | 3 | 5 | 7)
Positive-number ::= INTEGER (1 .. MAX) **Non-negative-number ::= INTEGER (0 | INCLUDES Postive-number)**
Date ::= SEQUENCE { **year INTEGER** **month INTEGER (1 .. 12)** **day INTEGER (1 .. 31)** **}**

Background to Exercise 1-3

```
Temperature   ::= [APPLICATION 0] REAL -- in ° Kelvin
WindVelocity  ::= [APPLICATION 1] REAL -- in m/s
Humidity      ::= [APPLICATION 2] REAL -- relative percent
Altitute      ::= [APPLICATION 3] INTEGER -- in meters

WeatherReading    ::= [APPLICATION 4] SEQUENCE
                  {   temperatureReading  Temperature,
                      velocityReading     WindVelocity,
                      humidityReading     Humidity,
                      altitude            Altitude }
```

Exercise 3

Change the definition of **Measurement** in Exercise 2 so that feet can only have the values 0, 1 or 2 (since 3 feet will be a yeard), and so that inches is specified as an integer between 0 and 1199 giving the value in hundreds of an inch (since 1200 or 12 inces will be a foot).

Background to Exercise 4-5

```
Person ::= SEQUENCE {   name  Name,
                        age  Integer,
                        agegroup Group }

Name ::= SEQUENCE {     givenname
                        GeneralString,
                        surname GeneralString }

Group ::= ENUMERATED { young (0),
                        middleaged (1),
                        old (2) }
```


Exercise 4

In an opinion poll, made outside the election rooms, every voter is asked to indicate which party they vote for. Allowed values are Labour, Liberals, Conservatives or “other”. The age of each voter is also registered as a positive integer above the voting age of 18 years, and the sex is registered. Define a data type to transfer this information from the poll station to a server.

Exercise 5

In the local election in Hometown, there are also two local parties, the Hometown party and the Drivers party. Extend solution 1 to exercise 4 to a new datatype **HometownVoter** where also these two additional parties are allowed.

Boolean — Simple Type

Värdemängd: TRUE och FALSE	
Typnotation: **BOOLEAN**	**Värdenotation:** **TRUE** **FALSE**
Subtyper: **Single value, Contained subtype**	

Anmärkning: Jag tycker det borde vara lagligt att skriva t.ex.

```
Sex ::= BOOLEAN {male (TRUE), female (FALSE) }
```

men det lär inte vara tillåtet.

Enumerated — Simple Type

Värdemängd: Varje uppräkning av skilda, namngivna värden	
Typnotation: **ENUMERATED { <ident> (<num>), ...}** **Exempel:** **ENUMERATED** **{ touristClass (-1),** **businessClass(0),** **firstClass(1)** **}**	**Värdenotation:** **<ident>** **Exempel:** **touristClass** **firstClass**
Subtyper: Single value, Contained subtype	

Tre möjliga notationer för veckodag


```
DayOfTheWeek    ::= INTEGER { monday(1), tuesday(2),
                      wednesday(3), thursday(4), friday(5),
                      saturday(6), sunday(7) }
DayOfTheWeek    ::= INTEGER { monday(1), tuesday(2),
                      wednesday(3), thursday(4), friday(5),
                      saturday(6), sunday(7) } (1..7)
DayOfTheWeek    ::= ENUMERATED { monday(1), tuesday(2),
                      wednesday(3), thursday(4), friday(5),
                      saturday(6), sunday(7) }
```

I det översta fallet tillåts alla heltal, i den mittersta fallet tillåts bara de sju heltalsvärdena från 1 till 7.

Skillnad mellan mittersta och nedre fallet: Ordning definierad för INTEGER, inte för ENUMERATED. Detta innebär att jämförelser med < och > och range-subtyper inte tillåts för Enumerated. Jämför programmeringsspråket Pascal.

Real — Simple Type

Värdemängd: $\pm\infty$ och heltal som kan uttryckas på formen M X B^E där Mantissan M kan vara godtycklig INTEGER, Basen B kan vara 2 eller 10, Exponenten E kan vara godtycklig INTEGER	
Typnotation: **REAL**	**Värdenotation:** **{ <num>, <num>, <num> }** **0** **PLUS-INFINITY** **MINUS-INFINITY** **Exempel:** **{ 3141592653589793238462433832 79, 10, -30}**
Subtyper: **Single value, Contained subtype, Range**	

Exempel på användning av Real

```
Temperature      ::= [ APPLICATION 0] REAL
                       -- In degrees Kelvin
pi REAL          ::= {31415926535879323846243 3, 10, 25 }
zero REAL        ::= 0
upperLimit REAL  ::= PLUS-INFINITY
```


Background to Exercise 6-7

```
Weekday ::= INTEGER { mon (1), tue(2),
            wed(3), thu(4), fri(5), sat(6), sun
            (7) } (1..7)

Weekday ::= ENUMERATED { mon (1),
            tue(2), wed(3), thu(4), fri(5),
            sat(6), sun (7) }
```


Exercise 6

In the armed forces, three degrees of secrecy are use: open, secret and top secret. Suggest a suitable datatype to convey the secrecy of a document which is transferred electronically.

Exercise 7

Given the solution to Exercise 6, assume that a new degree extra high secret is wanted. Define an extended version of the protocol defined in Exercise 6 to allow also this value.

Bit String — Simple Type

Värdemängd: Ordnad följd av 0 eller fler bitar	
Typnotation: BIT STRING BIT STRING { <ident> (<num>), ...} **Exempel:** BIT STRING { oddparity (0), enableparity (1), eightdatabits (2) }	**Värdenotation:** '<binära siffror>' B '>hexadecimala siffror>' H { <identifierare>, ... } **Exempel:** '0001010' B '00FF' H { oddparity, enableparity }
Subtyper: Single value, Contained subtype, Size range	

Anmärkning: Kodas enligt BER mer kompakt än SEQUENCE of BOOLEAN, men ej mer kompakt vid Packed Encoding Rules.

Subtyper till BITSTRING

Utöver Single value och Contained subtype finns även Size range. Exempel:

```
BIT STRING (SIZE ( 0 | 2 .. 7 | 10 ) )
```

Exempel på BITSTRING

```
BitMappedPicture    ::= BIT STRING
Characteristics     ::= BIT STRING {male(0), adult(1), blueEyed(2),
                           caucasian(3) }
Characteristics     ::= BIT STRING {male(0), adult(1), blueEyed(2),
                           caucasian(3) } (SIZE (0 .. 4))
Characteristics     ::= BIT STRING {male(0), adult(1), blueEyed(2),
                           caucasian(3) } (SIZE (4))
```

Vad är skillnaden mellan de tre definitionerna av Characteristics ovan?

Background to Exercise 8-9

```
Delivery    ::= [APPLICATION 4] SEQUENCE
                { price          Price,
                  weight         Weight,
                  weekday        Days }
Price       ::= [APPLICATION 0] REAL -- EURO
Weight      ::= [APPLICATION 2] REAL -- in
                  grams
Day         ::= [APPLICATION 5] BIT STRING {
                  mon(1), tue(2), wed(3), thu(4),
                  fri(5), sat(6), sun(7) } (SIZE(7))
```


Exercise 8

Assume that you want to define a pattern to cover a monochrome screen. Each pixel on the screen can be either black or white. The pattern is made by repeating a rectangle of N times M pixels over the whole screen. Examples of possible patterns are:

Base Example of use Base Example of use

Specify an ASN.1 data type which you can use to describe different such patterns.

Exercise 9

A store holds paper in the formats A3, A4, A5 and A6. A user wants to know if sheets are available in each of these four formats. Specify a data type to report this to the user.

Jämförelse av Bit String och Enumerated

```
DayOfTheWeek  ::= ENUMERATED { monday(0),
                    tuesday(1), wednesday(2),
                    thursday(3), friday(4), saturday(5),
                    sunday(6) } }

DaysOpen      ::= BIT STRING { monday(0), tuesday(1),
                    wednesday(2), thursday(3),
                    friday(4), saturday(5), sunday(6) }
                    (SIZE(7))
```

Vad betyder ”monday” i de två fallen ovan?

Octet String — Simple Type

Värdemängd: Ordnad följd av 0 eller fler oktetter	
Typnotation: OCTET STRING	**Värdenotation:** '<binära siffror>' B '>hexadecimala siffror>' H **Exempel:** '00001010' B '00FF' H
Subtyper: Single value, Contained subtype, Size range	

Exempel på Octet String

```
PackedBCDString      ::= OCTET STRING
-- the digits 0 through 9, two digits per octet,
-- each digit encoded as 0000 to 1001,
-- 1111 used for padding.

twelve  PackedBCDString::= '12'H
```

Null — Simple Type

Värdemängd: Ett enda värde: **null**	
Typnotation: **NULL**	**Värdenotation:** **NULL**
Subtyper: **Single value, Contained subtype**	
Exempel: **Order ::= SEQUENCE {** **ISBN VisibleString,** **Airmail NULL OPTIONAL }**	

Anmärkning: Kan användas för att markera plats för något som skall komma, eller när enbart existensen ger information, används sällan.

Exempel på användning av SIZE

MonthNumber ::= NumericString (SIZE (1 .. 2))
MonthNumber ::= NumericString (SIZE (1 | 2))
Base ::= BIT STRING (SIZE (0 | 2 .. 7 | 10))
Couple ::= SET SIZE(2) OF Human
BridgeDeal ::= SET SIZE (13) OF PlayingCard
BridgeHand ::= SET SIZE (0..13) OF PlayingCard

lineLength INTEGER 80
Line ::= VisibleString (SIZE (0 .. lineLength)

Background to Exercise 10

no-of-months INTEGER ::= 12

MonthsOpen ::= BIT STRING (SIZE (no-of-months))

Exercise 10

The X.400 standard specifies that a name can consist of several subfields. One of the subfields is called **OrganizationName** and can have as value between 1 and 64 characters from the character se **PrintableString**. Suggest a definition of this in ASN.1.

Character String-typer

Värdemängd: En sträng av tecken ur ett visst alfabet	
Typnotation: **NumericString** **PrintableString** **TeletexString** **T61String** **VideotexString** **VisibleString** **ISO646String** **IA5String** **GraphicString** **GeneralString** **UniversalString**	**Värdenotation:** **"<sträng>"** **Exempel:** **"PS example"** **"Alfvén"** **"αβχδεφγηιφκλ"**
Subtyper: Single value, Contained subtype, Size Range, Permitted alphabet (finns bara för teckensträngar)	

Alfabet för Character String-typerna

NumericString	'0' .. '9' och ' '
PrintableString	'a'..'z', 'A'..'Z', '0'..'9' ' ( ) + , - . / : = ?
TeletexString **T61String**	Se T.61, ca 400 tecken inklusive diakritiska tecken, t.ex. "ä" = "a with diaresises", klarar alla nationella varianter av latinska alfabetet
VideotexString	Se T.100 och T.101
VisibleString **ISO646String**	Tryckbara tecken plus blanksteg ur ISO 646 (”ascii”)
IA5String	IA5 (ISO 646, ”ascii”)
GraphicString	Alla hos ISO registrerade teckenmängder (G-mängderna) plus blanksteg. ISO 2022 escape sequences can switch between sets.
GeneralString	Alla hos ISO registrerade teckenmängder (G och C-mängderna) plus blanksteg och delete. ISO 2022 escape sequences can switch sets.
UniversalString	ISDO 10646 (Unicode)

Subtyp till Character String-typer

Permitted Alphabet, alla tillåtna tecken måste räknas upp, ingen ordning gäller.

Exempel:

PrintableString (FROM("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7"))

Set — Structured Type

Värdemängd: Värdet är en samling av värden från ett antal komponent-typer. Samlingen har ett värde för varje obligatorisk komponent, och noll eller ett för varje valfri komponent	
Typnotation: **SET { <component> , ... }** där **<component >**är någon av **• <identifier> <type>** **• <identifier> <type> OPTIONAL** **• <identifier> <type> DEFAULT <value>** **• COMPONENTS OF <type>**	**Värdenotation:** **{ identifier value,** . . . **}**

Exempel:	**Exempel:**
SET **{ day INTEGER** **name IA5String OPTIONAL** **}**	**{ day 12, name "Agneta" } - - *Rätt*** **{ name "Mary Anne", day 5 } - - *Rätt*** **{ day -4711 } - - *Rätt*** **{ 12, "Agneta" } - - *Fel***

Subtyper: Single value, Contained subtype, Inner subtyping (= subtyping av en eller flera av komponenterna, eller ändrad optionalitet)

Background to Exercise 11

UnicodeChar ::= SET {
description [1] PrintableString,
languages [2] SET OF PrintableString,
countries [3] SET OF PrintableString OPTIONAL,
hexcode [4] HexString }

HexString ::= PrintableString (FROM ("0" | "1" | "2" | "3" | "4"|"5" | "6" | "7" | "8" | "9"| "A" | "B"| "C" | "D"| "E" | "F") (SIZE(4))

Exercise 11

In a protocol for transferring personal data between two computers, a social security number is transferred. This number consists of only digits, blanks and dashes. Name (not split into first name and surname, max 40 characters) can also be transferred if known, and an estimated yearly income can be transferred if known. Both of these values are optional, only the social security number is mandatory. Specify using the **SET** construct of ASN.1 a datatype to transfer this information.

Exercise 12

Assume that a name is to be transferred as two fields, one for given name and one for surname. How can the solution to Exercise 11 be changed to suit this case?

Inner subtyping av Set-typen

(Single Inner Type: För SET OF och SEQUENCE OF)

WITH COMPONENT <subtype-spec>

(Multiple Inner Type for SET och SEQUENCE)

WITH COMPONENTS { <ident?> <subtype -spec?> <presence?>, . . . }
WITH COMPONENTS { ... , <ident?> <subtype-spec?> <presence?>, . . . }

De första tre punkterna ingår i ASN.1-språket och anger att alla övriga komponenter ingår fast de inte räknas upp.

Avslutande tre punkter ingår ej i ASN.1-språket.

Presence kan ha värdena **PRESENT, ABSENT, OPTIONAL**

Exempel på Inner subtyping av Set-typen


```
DoubleFormatName ::= SET
{       psform PrintableString OPTIONAL,
        t61form TeletexString OPTIONAL}

OnlyPSform ::= DoubleFormatName
        (  WITH COMPONENTS
           { ... , t61form ABSENT } )

OctalPSform ::= DoubleFormatName
        (  WITH COMPONENTS
           { psform (FROM( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" ) )
             t61form (FROM( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" ) )
           }
        )
```


WITH COMPONENTS, exempel


```
NormalName ::= SEQUENCE {
        givenName       [0] GraphicString OPTIONAL,
        surName         [1] GraphicString OPTIONAL,
        generation      [2] GraphicString OPTIONAL,
        age [3] INTEGER
        }
RoyalName ::= NormalName
        ( WITH COMPONENTS {
                givenName PRESENT,
                surName ABSENT,
                generation PRESENT
                age (18.. MAX) }
        )
```

Exercise 13

Given the ASN.1-type:

```
XYCoordinate ::= SEQUENCE {
                        x REAL,
                        y REAL
                        }
```

Define a subtype which only allows values in the positive quadrant (where both x and y are >= 0).

Exercise 14

Given the ASN.1 type:

```
Message ::= SET {
            author Name OPTIONAL,
            textbody IA5String }
```

Define a subtype to this, called **AnonymousMessage**, in which no **author** is specified.

Exempel 1 på värde-notation för Set-typen

```
Building::= SET
{       address OCTET STRING,
        occupied BOOLEAN
}

headquarters  Building ::=
{       address '313434346E45765433DA0' H,
        occupied TRUE
}
```


Exempel 2 på värde-notation för Set-typen

```
PersonnelRecord ::= [0] IMPLICIT SET
{       name [0] IMPLICIT OCTET STRING,
        Location [1] IMPLICIT INTEGER
        { homeOffice(0), fieldOffice(1), roving (2) } OPTIONAL,
        age [2] IMPLICIT INTEGER OPTIONAL
}

director PersonnelRecord ::=
{       location homeOffice,
        name '44578E4F3A0F' H,
        age 44
}
```


Sequence — Structured Type

Värdemängd: varje värde är en samling av värden från ett antal komponent-typer. Samlingen måste vara ordnad, och har ett värde för varje obligatorisk komponent och noll eller ett värde för varje valfri komponent	
Typnotation: **SEQUENCE { <component,> ...}** där <**component**> kan ta samma värden som SET-typen	**Värdenotation:** **{<identifier> <value>,** **. . .** **}**

Sequence (forts.)

Exempel:	**Exempel:**
SEQUENCE **{ day INTEGER** **name IA5String** **OPTIONAL** **}**	**{ day 12, name "Agneta" }** **{ day 5 }** **{ day -4711 }** **{ 5, "Mary Anne" }** **{ name "Jean" day 17 } - - *Fel*** ***- - ordningen måste stämma!***
Subtyper: Single value, Contained subtype, Inner subtyping	

Observera att komponentnamnen inte behöver anges i värdenotationen, se det andra exemplet ovan.

Exercise 15

Define a datatype **FullName** which consists of three elements in given order: Given name, Initials and Surname. Given name and Initials are optional, but Surname is mandatory.

Set-of — Structured Type

Värdemängd: Varje värde är en oordnad mängd av värden av en viss känd typ	
Typnotation: **SET OF <type>** **SET <size-limit> OF <type>** **Exempel:** **SET OF Name** **där** **Name ::= SEQUENCE** **{ GivenName IA5String,** **SurName IA5String }**	**Värdenotation:** **{ <value>, . . . }** **Exempel:** **{ { "John", ”Green” },** **{ "Mary", "Green" },** **{ ”John”, ”Green” }** **}**
Subtyper: Single value, Contained subtype, Size range, Inner subtyping	

Sequence of — Structured Type

Värdemängd: Varje värde är en ordnad mängd av värden av en viss känd typ	
Typnotation: **SEQUENCE OF <type>** **SEQUENCE <size-limit> OF <type>** **Exempel:** **SEQUENCE OF City** **där City ::= SEQUENCE {** **name IA5String,** **longitude INTEGER,** **latitude INTEGER }**	**Värdenotation:** **{ <value>, . . . }** **Exempel:** **{ { "Stockholm", 59 ,18 },** **{ "London", 51, 0 },** **{ "Berlin", 52, 13 },** **{ "Stockholm", 59, 18 } }** **{ }**
Subtyper: Single value, Contained subtype, Size range, Inner subtyping	

Exercise 16

Define a data type **BasicFamily** consisting of 0 or 1 **husband**, 0 or 1 **wife** and 0, 1 or more **children**. Each of these components are specified as an **IA5String**.

Exercise 17

Define a datatype **ChildLessFamily**, based on **BasicFamily** from Exercise 16.

Choice — Structured Type

Värdemängd: Summan av värdena för alla komponenttyperna	
Typnotation: CHOICE { <ident> <type>, . . . } **Exempel:** CHOICE { arabicNumber NumericString, romanNumber PrintableString }	**Värdenotation:** *(1988 års version)* <ident> <value> *(1992 års version)* <ident> : <value> **Exempel:** arabicNumber "6" - - *1988* romanNumber "VI" - - *1988* romanNumber : "VI" - - *1992*
Subtyper: Single value, Contained subtype, Inner subtyping	

Exempel 1 på värde-notation för Choice-typen

```
MessageType ::= CHOICE
{                   text OCTET STRING,
                    codedNumeric INTEGER
}

initialize  MessageType  ::=  text '0000000000000000'B
panic MessageType  ::=  codedNumeric 13
```

Exempel 2 på värde-notation för Choice-typen


```
Division  ::=  CHOICE
{           manufacturing    [0] IMPLICIT SEQUENCE
            {                plantID INTEGER,
                             majorProduct OCTET STRING
            },
            r-and-d          [1] IMPLICIT SEQUENCE
            {                labID INTEGER,
                             currentProject OCTET STRING
            } }
currentAssignment  Division ::=
            r-and-d
            {                labID 48,
                             currentProject '4458D37' H }
```

Selection type — Structured Type

Värdemängd: Utgår från en CHOICE-typ, väljer ut ett av alternativen

Typnotation:	**Värdenotation:**
identifer < <type>	*(Samma som för den utvalda typen)*
Exempel:	**Exempel:**
ArabicForm ::= arabicNumber < GeneralNumber	"6" "4711"
GeneralNumber::= CHOICE { arabicNumber NumericString, romanNumber PrintableString }	OBS: **a < x** och **x (WITH COMPONENTS {a})** är två sätt att uttrycka samma sak

Subtyper: Single value, Contained subtype, Inner subtyping

Background to Exercise 18

```
Co-ordinates ::=      CHOICE { cartesian
                               Cartesian-co-ordinates,
                               polar Polar-co-ordinates }

Cartesian-co-ordinates ::= SEQUENCE {
                               x REAL,
                               y REAL }

Polar-co-ordinates ::= SEQUENCE {
                               radius REAL,
                               angel REAL }
```


Exercise 18

Given the data types **Aircraft**, **Ship**, **Train** and **MotorCar**, define a datatype **Vessel** whose value can be any of these datatypes.

Exercise 19

What is the difference between the data type:

```
NameListA ::= CHOICE {
                    ia5 [0] SEQUENCE OF IA5String,
                    gs [1] SEQUENCE OF GeneralString
                    }
```

and the data type:

```
NamelistB ::= SEQUENCE OF CHOICE {
                    ia5 [0] IA5String,
                    gs [1] GeneralString
                    }
```

How is it in both alternatives above possible to define a new data type **GeneralNameList** which only can contain a **GeneralString** element?

Background to Exercise 20

Tyres ::= CHOICE { bike Biketyres,
car (Cartyres }

Biketyres ::= SEQUENCE SIZE (2) OF Tyre

Cartyres ::= SEQUENCE SIZE(4) OF Tyre

Exercise 20a

The by-laws of a society allows two kinds votes:

(a) The voters can select one and only one of 1 .. N alternatives. The alternative which gets the most total votes wins.

(b) The voters can indicate a score of between 0 and 10 for each of the choices 1 .. N. The choice which gets highest total score wins.

Specify an ASN.1 data type which can be used to report the votings of a person to the vote collection agent, and which can be used for both kinds of votes. The name of the voter shall be included in the report as an **IA5String**.

Exercise 20b

Suggest a textual encoding for Exercise 20a using ABNF.

Any — Structured Type

Värdemängd: Alla värden hos alla typer	
Typnotation: **ANY** **ANY DEFINED BY <identifier>** <identifier> kan vara t.ex. ett heltal eller en objektidentifierare	**Värdenotation:** **(1988) <type> <value>** **(1992) <type> : <value>**
Exempel: **SEQUENCE** **{ type INTEGER,** **value ANY DEFINED BY type }**	**Exempel (1988 års notation):** **BOOLEAN TRUE** **BIT STRING '101' B**
Subtyper: Single value, Contained subtype	

Kan typen härledas ur sammanhanget?


```
Name      ::= SEQUENCE
          {   givenName       [0]  VisibleString OPTIONAL,
              surName         [1]  VisibleString OPTIONAL }
Name      ::= SET
          {   givenName       [0]  VisibleString,
              surName         [1]  VisibleString }
Name      ::= CHOICE
          {   numericName     NumericString,
              alphabeticName  VisibleString }
```

Alla alternativ måste ha olika typer för:

- Komponenter i ett SET
- Komponenter i en SEQUENCE med OPTIONAL
- Komponenter i en CHOICE

Om bastyper inte är olika, kan de göras olika med etiketter (tags)

OBS att man ändå kan ta bort de två förekomsterna av **[0]** i exemplet ovan, därför att då får det ena objektet Universal-taggen för VisibleString, det andra context-taggen **[1]** och det räcker för att de skall anses vara olika.

Etiketter (Tags)

Tags (etiketter) används för att skilja på olika typer. Tags är nödvändiga i sådana situationer där typen inte framgår av sammanhanget, t.ex. i en oordnad, blandad mängd av objekt av olika typer. Men tags får användas även när det inte är absolut nödvändigt, och det anses numera vara god ASN.1 att använda tags även när det inte är nödvändigt.

Orsak: Om man har ett element med en Tag, så kan man i framtida nya versioner av protokollet tillåta nya värden med andra Tags. Har man ingen Tag på ett element, får man det mycket svårare när man i framtiden skall definiera en utvidgad version av ett protokoll.

En Tag består av två komponenter:

- Tag class
- Tag number

Fyra klasser av etiketter(tags)

Klass	Exempel	Beskrivning
Universal	**[UNIVERSAL 1]**	I ASN.1-standarden definierad tag. [Universal 1] är t.ex. definierad som standard-tag för typen Boolean.
Application	**[APPLICATION 3]**	Har samma betydelse överallt inom en applikations-modul.
Private	**[PRIVATE 4]**	Om ett företag eller en organisation vill göra egna utvidgningar av ASN.1
Context	**[7]**	En omgivningsberoende (context dependent) tag har sitt värde bara i just det sammanhang där den används.

Anmärkning 1: I 1994 års ASN.1 avråds från användning av Application och Private tags.

Anmärkning 2: Med Automatic tagging behöver man sällan ange taggar.

Exempel på omgivningsberoende (context-dependent) tags:


```
Name           ::= SET {
                       given name    [0] VisibleString,
                       surname       [1] VisibleString }

PersonellRecord ::= SET {
                       name          [0] Name,
                       wage          [1] INTEGER }
```

I detta exempel betyder tag-värdena [0] och [1] olika saker i de två exemplen på ASN.1-produktioner. I det övre fallet betyder [1] efternamn, i det undre fallet betyder [1] ett lönefält.

Vilka Tag-klasser används mest?

UNIVERSAL	Kräver att de i standarden inbyggda typerna räcker.
APPLICATION	Ger problem vid export och import. Ej rekommenderad i 1994 års ASN.1.
PRIVATE	Ger problem vid hopkoppling av olika tillämpningar, ANY och EXTERNAL kan göra samma sak bättre. Ej rekommenderad i 1994 års ASN.1.
CONTEXT	Entydiga i givet sammanhang.
(AUTOMATIC	Görs om till CONTEXT eller UNIVERSAL alltefter behov)

I ASN.1 fördefinierade etiketter: Universal Tags

Simple types

1	BOOLEAN
2	INTEGER
3	BIT STRING
4	OCTET STRING
5	NULL
6	OBJECT IDENTIFIER
9	REAL
10	ENUMERATED

Structured types

16	SEQUENCE
16	SEQUENCE OF
17	SET
17	SET OF
(a)	CHOICE
(b)	ANY

(a) = Alla tags för de tillåtna alternativen
(b) = Samtliga möjliga tags

Character String Types

12	UTF8String
18	NumericString
19	PrintableString
20	TeletexString
21	VideotexString
22	IA5String
25	GraphicString
26	VisibleString
27	GeneralString
28	UniversalString
29	CharacterString
30	BMPString

UsefulTypes

7	ObjectDescriptor
8	EXTERNAL
23	UTCTime
24	GeneralizedTime

Tagged — Structured Type

Värdemängd: Samma som någon annan typ, men med en ny, användargiven Tag

Typnotation:

[<tagclass> <tagnumber>] <type>
[<tagclass> <tagnumber>] IMPLICIT <type>
[<tagclass> <tagnumber>] EXPLICIT <type>

där **<tagclass>** kan vara
UNIVERSAL, APPLICATION eller
PRIVATE
om **<tagclass>** inte anges antas
"Context-specific"

Exempel:

[APPLICATION 7] INTEGER (0..9)
[0] IMPLICIT REAL
[PRIVATE 7] EXPLICIT BOOLEAN

Värdenotation:

Samma som för den underliggande typen

IMPLICIT och EXPLICIT får bara anges direkt efter en tag, och anger om den nya taggen skall ersätta eller komplettera tag för underliggande typ. Om ingendera anges antas förvalt värde för denna modul.

Subtyper: Samma som för underliggande typ

Explicit och Implicit tags

I modulhuvudet kan man ange

DEFINITIONS ::=
DEFINITIONS IMPLICIT TAGS ::=
DEFINITIONS EXPLICIT TAGS ::=
DEFINITONS AUTOMATIC TAGS ::= (I 1994 års ASN.1)

Om varken IMPLICIT TAGS eller EXPLICIT TAGS anges antas EXPLICIT TAGS. AUTOMATIC TAGS innebär EXPLICIT för CHOICE and Open Types, IMPLICIT otherwise.

Man kan sedan i texten ange t.ex.

Height [0] IMPLICIT REAL
Height [0] EXPLICIT REAL
Height [0] REAL

När varken IMPLICIT eller EXPLICIT anges gäller förval för hela modulen, enligt modulhuvudet.

Example of the same module specified with IMPLICIT and EXPLICIT tags

PersonnelFile { 2 3 4 4711 6 } **DEFINITIONS IMPLICIT TAGS ::=**	**PersonnelFile { 2 3 4 4711 6 }** **DEFINITIONS EXPLICIT TAGS ::=**
BEGIN	**BEGIN**
PersonRecord ::= SET { **name IA5String,** **wage [0] EXPLICIT INTEGER,** **age [1] INTEGER }**	**PersonRecord ::= SET {** **name IA5String,** **wage [0] INTEGER,** **age [1] IMPLICIT INTEGER }**
Person ::= [APPLICATION 1] **PersonRecord**	**Person ::= [APPLICATION 1]** **IMPLICIT PersonRecord**
Robot ::= [APPLICATION 2] **EXPLICIT PersonRecord**	**Robot ::= [APPLICATION 2]** **PersonRecord**
END - - of module PersonnelFile	**END - - of module PersonnelFile**

Exercise 21

Assume an ASN.1-module which looks like shown below; Change this ASN.1 module, so that the same coding is specified, but with tag defaults **IMPLICIT** instead of **EXPLICIT**.

```
WeatherReporting {2 6 6 247 1} DEFINITIONS EXPLICIT TAGS ::=
BEGIN
WeatherReport ::= SEQUENCE {
                        height [0] IMPLICIT REAL,
                        weather [1] IMPLICIT Wrecord
                    }
Wrecord ::= [APPLICATION 3] SEQUENCE {
                        temp Temperature,
                        moist Moisture
                        wspeed [0] Windspeed OPTIONAL
                    }
Temperature ::= [APPLICATION 0] IMPLICIT REAL
Moisture ::= [APPLICATION 1] REAL
Windspeed ::= [APPLICATION 2] REAL
END - - of module WeatherReporting
```

Which tags can be removed?

Record ::= SEQUENCE { **GivenName [0]** **PrintableString** **SurName [1] PrintableString }**	Both tags can be removed
Record ::= SET { **GivenName [0]** **PrintableString** **SurName [1] PrintableString }**	One of the tags can be removed
Record ::= SEQUENCE { **GivenName [0]** **PrintableString OPTIONAL** **SurName [1] PrintableString** **OPTIONAL }**	One of the two tags can be removed

Exercise 22

Which tags below can be removed in correct ASN.1, not using automatic tagging?

```
Colour ::= [APPLICATION 0] CHOICE {
                        rgb [1] RGB-Colour,
                        cmg [2] CMG-Colour,
                        freq [3] Frequency
                        }
RGB-Colour ::= [APPLICATION 1] SEQUENCE {
                        red [0] REAL,
                        green [1] REAL OPTIONAL,
                        blue [2] REAL
                        }
CMG-Colour ::= SET {    cyan [1] REAL,
                        magenta [2] REAL,
                        green [3] REAL
                        }
Frequency ::= SET {     fullness [0] REAL,
                        freq [1] REAL }
```


Exercise 23a

The following ASN.1 construct is taken from the 1988 version of the X.500 standard. (**OPTIONALLY-SIGNED** is a macro, macros were replaced with a new construct in the 1994 version of ASN.1.)

```
ListResult ::= OPTIONALLY-SIGNED
        CHOICE {
        listInfo SET {
        DistinguishedName OPTIONAL,
        subordinates [1]SET OF SEQUENCE {
        RelativeDistinguishedName,
        aliasEntry [0] BOOLEAN DEFAULT FALSE
        fromEntry [1] BOOLEAN DEFAULT TRUE},
        partialOutcomeQualifier [2]
        PartialOutcomeQualifier OPTIONAL
        COMPONENTS OF CommonResults },
        uncorrelatedListInfo[0] SET OF ListResult }
```


Exercise 23b

(b) Is there anything wrong in the ASN.1 code??

Exercise 23c

(c) Why is there no identifier on the element COMPONENTS OF?
What does it mean?

Exercise 23d

(d) Why is there no context-dependent tags on some of the elements, but not on all of them?

GeneralizedTime

Datum och tidpunkt med olika precision enligt ISO-standarder. Formatet är i huvudsak

YYYYMMDDHHMMSS.SSS±HHMM eller YYYYMMDDHHMMSS.SSSZ

Samma tidpunkt, 5 minuter och 33,8 sekunder efter 7 på morgonen den 2 januari 1982 i New York City kan anges som

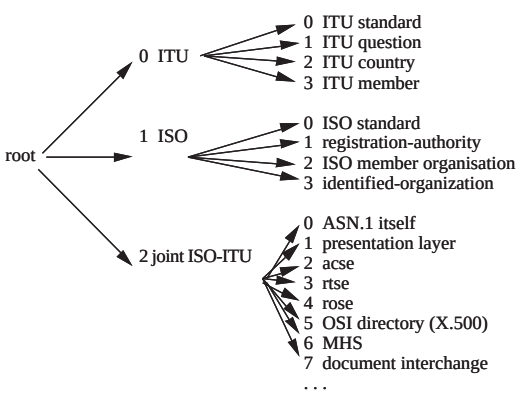
eventtime GeneralizedTime ::= "19820102070533.8" eller
eventtime GeneralizedTime ::= "19820102070533.8Z" eller
eventtime GeneralizedTime ::= "19820102070533.8-0500"

UTCTime (Universal time)

Något enklare och mer kortfattat än Generalized time. Årtal anges med bara två siffror, tiden anges i antingen hela minuter eller hela sekunder. Exempel:

eventtime UTCTime ::= "820102070534-0500"

Object identifier



Object identifier — Useful Type

Värdemängd: Noderna i objekt-identifierarträdet		
Typnotation: OBJECT IDENTIFIER	**Värdenotation:**	{ <linje> . . . } eller { <värdereferens> <linje> . . . } där <linje> kan vara <identifierare>, <nummer> eller <identifierare>(<nummer>)
	Exempel:	{joint-iso-ccitt ds (5) attributeTypes (4) telephoneNumber (14) } { 2 5 4 14 } { attributeType 14 }
Subtyper: Single value, Contained subtype		

Object Descriptor — Useful Type

ObjectDescriptor ::= [UNIVERSAL 7] IMPLICIT GraphicString

Används mest tillsammans med objekt-identifierare, för att erbjuda en för människor begriplig beskrivning av det som objekt-identifieraren identifierar.

External — Useful Type

Liknar typen Any, men en External kan innehålla ett värde som inte är i ASN.1-format. Externals yttre format kan definieras i ASN.1 på följande sätt:

```
EXTERNAL ::= [ UNIVERSAL 8 ] IMPLICIT SEQUENCE
{   direct-reference          OBJECT-IDENTFIER   OPTIONAL,
    indirect-reference        INTEGER            OPTIONAL,
    data-value-descriptor     ObjectDescriptor   OPTIONAL,
    encoding  CHOICE
    {       single-ASN1-type    [0]   ANY,
            octet-aligned       [1]   IMPLICIT OCTET STRING,
            arbitrary           [2]   IMPLICIT BIT STRING
    }
}
```

Minst en av tre OPTIONAL-alternativen måste ha ett värde.
Innehållets typ kan alltså anges antingen genom en OBJECT-IDENTFIER (direct-reference) eller genom en INTEGER (indirect-reference). I det senare fallet tilldelas detta heltal ett värde i presentationslagret.

Moduler

En modul är en namngiven samling av ASN.1 typdefinitioner och värdedefinitioner.

Notation:

```
<moduleReference>    <obj-id > DEFINITIONS <tag-defaults> ::=
BEGIN
    EXPORTS <type and value references>;
    IMPORTS <type and value references>
                                FROM <moduleReference> <obj-id>;
    ...
    <type and value definitions>
    ...
END
```

Om samma identifierare importeras från flera olika moduler, eller används både importerat och internt, kan man använda notationen:

```
modulereference.typereference
modulereference.valuereference
```


Exempel på modulnotation med IMPORTS

```
CargoHandling { 1  2  4711  17 } DEFINITIONS EXPLICIT TAGS ::=
BEGIN
EXPORTS Box, Container ;
Box ::= SEQUENCE {
                        height INTEGER, - -  in centimeters
                        width INTEGER, - -  in centimeters
                        length INTEGER } - -  in centimeters
Container ::= SEQUENCE
                        weight INTEGER, - -  in kilograms
                        volume Box }
END - -  of CargoHandling

TrainCargo { 1 2 4711 18 } DEFINITIONS EXPLICIT TAGS ::=
BEGIN
IMPORTS Box, Container FROM CargoHandling { 1  2  4711  17 };
TrainContainer ::= Container
                        ( WITH COMPONENTS
                                {    weight ( 0 .. 5000 ), volume }
                        )
Carriage ::= SET SIZE (2..4) OF Container
END - -  of TrainCargo
```


Exempel på modulnotation med punktnotation

```
CargoHandling { 1  2  4711  17 } DEFINITIONS EXPLICIT TAGS ::=
BEGIN
EXPORTS Box, Container ;
Box ::= SEQUENCE {
                        height INTEGER, - -  in centimeters
                        width INTEGER, - -  in centimeters
                        length INTEGER } - -  in centimeters
Container ::= SEQUENCE
                        weight INTEGER, - -  in kilograms
                        volume Box }
END - -  of CargoHandling

TrainCargo { 1 2 4711 18 } DEFINITIONS EXPLICIT TAGS ::=
BEGIN
Container ::= CargoHandling{ 1  2  4711  17 }.Container
                        ( WITH COMPONENTS
                                {    weight ( 0 .. 5000 ), volume }
                        )
Carriage ::= SET SIZE (2..4) OF Container
END - -  of TrainCargo
```


Exercise 24

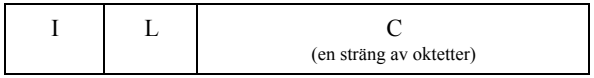
Given the following ASN.1 module:

```
Driving {1 2 4711 17} DEFINITIONS EXPLICIT TAGS ::=
                        BEGIN
MainOperation ::= SEQUENCE {
                        wheel [0] REAL,
                        brake [1] REAL,
                        gas [2] REAL  }
END
```

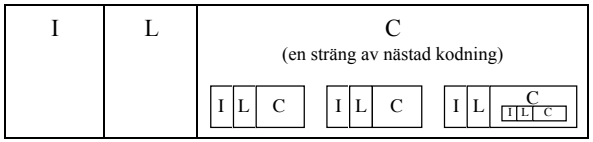
Define an ASN.1 module **CarDriving**, which imports **MainOperation** from the module above, and defines a new datatype **FullOperation** which in addition to **MainOperation** also includes switching on and of the left and right blinking lights, and setting the lights as unlit, parking lights, dimmed light and full beam.

Basic Encoding Rules (BER)

Primitive:

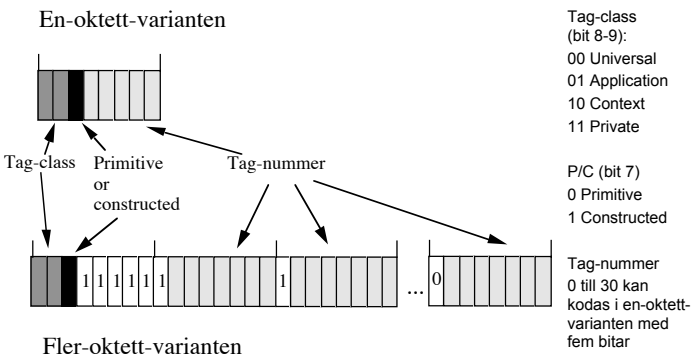


Constructed:

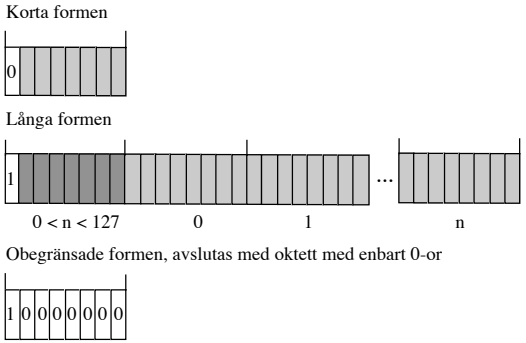


I = Identifier octets
L = Length octets
C = Contents octets

Identifierar-fältet i BER



Längd-fältet i BER



Avslutande av obegränsade formen

Observera att 0-oktetter mycket väl kan finnas inuti I, L och C-fältena i de underordnade fälten. Det är bara när avtolkaren väntar sig ett I-fält på samma nivå som startfältet, som avslutning sker med en 0-oktett. Eftersom tag-värdet 0 är reserverat för detta ändamål, kan en 0-oktett bara finnas när den avser en avslutning av den obegränsade formen.

Contents Octets

Boolean	En enda oktett. FALSE = 00000000 TRUE = alla andra värden
Integer	Tvåkomplementform, kodat i minsta nödvändiga antal oktetter.
Enumerated	Som för Integer.
Null	Ingen Contents Octet alls.
Object Identifier	A packet sequence of integers. Första motsvarar de första två linje-etiketterna, därefter en integer per etikett.
Set, Sequence, Set-of, Sequence-of	Nästade kodningar av komponenterna. Ordning är signifikant för sequence och sequence-of, inte för set och set-of
Choice, Any	Samma kodning som för utvald typ och värde

Contents Octets: Real

Fyra varianter:

- Noll representeras av ingen contents octet
- 01000000 för PLUS-INFINITY och 01000001 för MINUS-INFINITY
- Binärkodning med basen 2, 8 eller 16
- Decimalkodning enligt ISO 6093

I de tre nedre fallen anger första oktetten vilken kodningstyp som används.

Contents Octets: String

I primitive form: Bitarna, oktetterna eller de kodade tecknen utom för Bit String, där första oktetten anger hur många bitar i sista oktetten som skall ignoreras.

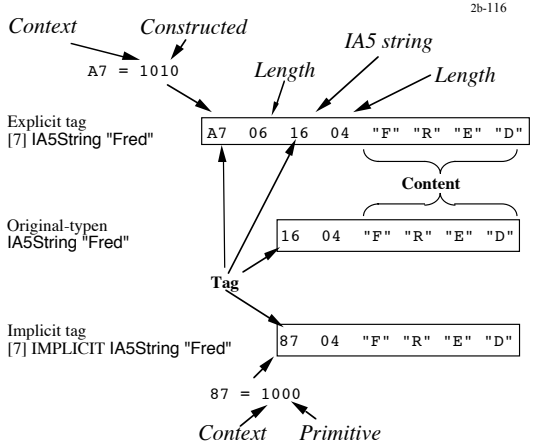
I constructed form, som om ASN.1 hade haft definitionerna:

BIT STRING ::= [UNIVERSAL 3] IMPLICIT SEQUENCE OF BIT STRING

OCTET STRING ::= [UNIVERSAL 4] IMPLICIT SEQUENCE OF OCTET STRING

IA5String ::= [UNIVERSAL 22] IMPLICIT SEQUENCE OF OCTET STRING

Implicit och explicit Tagging



Exempel på kodning av en SEQUENCE

```
HeadOfState ::= [APPLICATION 17] SEQUENCE
{       name IA5 STRING,
        type ENUMERATED {   president (0),
                            kejsare(1),
                            kung(2) }
        birthyear INTEGER OPTIIONAL }

swedishKing ::= {
        name "Carl XVI Gustav",
        type kung,
        birthyear 1946 }
```

$22_{10} = 16_{16}$= class universal(00), form primitive(0), tag number IA5(22)

0	0	0	1	0	1	1	0

30	18	Hexadecimala tal

16	0F	C	a	r	l		X	V	l		G	u	s	t	a	v

0A	01	02

02	02	1E	14

Exercise 25

Given the ASN.1 definition

```
Surname ::= [APPLICATION 1] IA5String
hername Surname ::= "Mary"
```

Show its coding in BER.

Exercise 26

Given the ASN.1 definition

```
Light ::= ENUMERATED {
                        dark (0),
                        parkingLight (1),
                        halfLight (2),
                        fullLight (3) }
daylight Light ::= halflight
```

give a BER encoding of this value.

Exercise 27

Given the following ASN.1 defintions and explicit tags

```
BreakFast ::= CHOICE {
        continental [0] Continental,
        english [1] English,
        american [2] American  }

Continental ::= SEQUENCE {
        beverage [1] ENUMERATED {
        coffea (0), tea(1), milk(2), chocolade (3) } OPTIONAL,
        jam [2] ENUMERATED {
        orange(0), strawberry(1), lingonberry(3) } OPTIONAL  }

English ::= SEQUENCE {
        continentalpart Continental,
```

Continued on the next slide

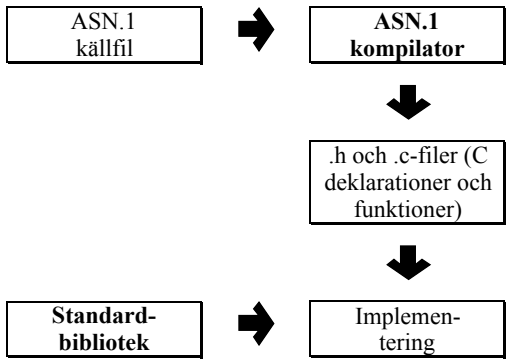
```
        eggform ENUMERATED {
        soft(0), hard(1), scrambled(2), fried(3)  }
Order ::= SEQUENCE {
        customername IA5String,
        typeofbreakfast Breakfast  }
firstorder Order ::= {
        customername "Johan",
        typeofbreakfast {
                english {
                continentalpart {
                        beverage tea,
                        jam orange
                        }
                eggform fried
                } } }
```

Give an encoding of **firstorder** with BER.

Alternativa kodningar

BER = Basic Encoding Rules	Not very efficient, much redundancy, good support for extensions
DER = Distinguished Encoding Rules	No encoding options (for security hashing), always use definite length encoding
CER = Canonical Encoding Rules	No encoding options (for security hashing), always use indefinite length encoding
PER = Packed Encoding Rules	Very compact, less extensible
LWER = Light Weight Encoding Rules	Almost internal structure, fast encoding/deconding

ASN.1-kompilatorer



Exempel på ASN.1 (X.420 sid 548-551)

```
IPM ::= SEQUENCE {
                heading     Heading
                body        Body }

Heading ::= SET {
                this-IPM                ThisIPMField,
                originator              [0] OriginatorField OPTIONAL,
                authorizing-users       [1] AuthorizingUsersField OPTIONAL,
                primary-recipients      [2] PrimaryRecipientsField DEFAULT {},
                copy-recipients         [3] CopyRecipientsField DEFAULT {},
                . . .
```

Fråga: Varför är det ingen TAG angiven för This-IPM ovan???

```
ThisIPMField ::= IPMIdentifier

IPMIdentifier ::= [APPLICATION 11] SET {
                user                    ORAddress OPTIONAL,
                user-relative-identifier    LocalIPMIdentifier }

LocalIPMIdentifier ::= PrintableString (Size (0..ub-local-ipm-identifier))
```

Exempel på ASN.1 (X.411 sid 336)


```
MessageSubmissionEnvelope ::= SET {
        COMPONENTS OF PerMessageSubmissionFields,
        per-recipient-fields [1] SEQUENCE SIZE (1..ub-recipients) OF
        PerRecipientMessageSubmissionFields }

PerMessageSubmissionFields ::= SET {
        originator-name OriginatorName,
        original-encoded-information-types
                OriginalEncodedInformationTypes OPTIONAL,
        content-type ContentType,
        priority Priority DEFAULT normal,
        per-message-indicators PerMessageIndicators DEFAULT {},
        deferred-delivery-time [0] DeferredDeliveryTime OPTIONAL,
        extensions [2] PerMessageSubmissionExtensions DEFAULT {} }
```


Makros i ASN.1

- Makros innebär ingen utvidgning av BER.
- Makros kan inte säkert expanderas till vanlig ASN.1.
- Makros kan alltid kodas med hjälp av BER.

Nyheter i 1994 års version av ASN.1

ASN.1	Del 1:	Basic Notation	Smärre nyheter
ASN.1	Del 2:	Information Object Specification	Ersätter Macro-notation
ASN.1	Del 3:	Constraint Specification	Hur man begränsar värdemängden för en ASN.1-syntax
ASN.1	Del 4:	Parameterisation of ASN.1 Specifications	Hur man kan definiera fack för framtida eller lokala utvidgningar
Encoding rules	Del 2:	Packed Encoding Rules	Ger mer kompakt kodning än ”Basic Encoding Rules”

Ny syntax i Basic Notation 1994

... A B * ...	är ekvivalent med	... C ...	där C::= empty | D D::= A | A B D
... A B + ...	är ekvivalent med	... E ...	där E::= A | A B E
... A ? ...	är ekivalent med	... F ...	där F::= empty | A

Type compatibility i Basic Notation 1994

Type equivalence

```
A ::= INTEGER(0..<25)
B ::= INTEGER(0..10|11..<25)
```

Subtyping

```
A ::= OCTET STRING
B ::= OCTET STRING
        (SIZE(1..10))
not  C:= [0] OCTET STRING
        (SIZE(1..10))
```

Type isomorphism

```
B ::= OCTET STRING
        (SIZE(1..10))
C:= [0] OCTET STRING
        (SIZE(1..10))
```

Subtype compatibility

```
A ::= OCTET STRING
B ::= OCTET STRING
        (SIZE(1..10))
C:= [0] OCTET STRING
        (SIZE(1..10))
```


Type Compatibility

```
A ::= INTEGER (0..99)
B ::= [0] INTEGER (50..999)
C ::= INTEGER (-10..<0)
```

A, B och C är alla Type compatible men inte Type equivalent eller Type isomorphic

Assignement-Compatibility

```
A ::= INTEGER (0..99)
a A ::= 60
B ::= [0] INTEGER(0..<10)
C ::= [1] INTEGER(50..999)
```

A, B och C är type compatible

B är sub-type compatible with A

a är assignment compatible with C men inte med B

Ny typ UNIVERSAL STRING

Motsvarar teckenstandarden ISO 10646

Ny typ CHARACTER STRING

Unrestricted character string, kan innefatta många olika existerande och framtida teckensträngstandarder

```
CHARACTER STRING ::= [UNIVERSAL 29] IMPLICIT SEQUENCE
        { syntax-id CHOICE
                { explicit SEQUENCE
                        {abvstract-syntax       OBJECT IDENTIFIER,
                        tramsfer-syntax         OBJECT IDENTIFER},
                defined-context       INTEGER},
        string-value OCTET STRING }
```

(behöver ihte kodas på det sättet, men tagen är **UNIVERSAL 29**)

Ny datatyp: Embedded PDV

Utvidgning av EXTERNAL-typen

```
EXTERNAL PDV ::= [UNIVERSAL 11] IMPLICIT SEQUENCE
{
        syntax-id CHOICE
        {
                explicit SEQUENCE
                {
                        abstract-syntax OBJECT IDENTIFIER,
                        transfer-syntax OBJECT IDENTFIER
                },
                defined-context INTEGER
        },
        pdv-value BIT STRING
}
```

(behöver inte kodas på det sättet, men tagen är **UNIVERSAL 11**)

Information Object Specification

Ersättning för Macro-faciliteten i 1988 års ASN.1

An information object class utmärks av de typer av fält som instanser av den kan ha. Dessa kan vara

- An arbitrary type (a type field)
- A single value of a specified (a fixed-type value field)
- A single value of a type specified in a named type field (a variable-type value field)
- An arbitrary non-empty set of values of a specified type (a value set field)
- A single information object from a specified information object class (an object-field)
- An information object set from a spåecified information object class (an object set field)

Man specificerar en *information object class* genom att specificera:

- Namnen på fälten
- För varje fält, fältets typ
- Om fältet är OPTIONAL eller DEFAULT
- Om något fält är identifierar-fältet (UNIQUE)

Exempel på Information Object Class

Definition av en ROS-liknande operation på det nya sättet:

```
OPERATION  ::=  CLASS
{
        &ArgumentType OPTIONAL,
        &ResultType       OPTIONAL,
        &Errors           ERROR OPTIONAL,
        &Linked           OPERATION OPTIONAL,
        &resultReturned BOOLEAN DEFAULT TRUE,
        &code             INTEGER UNIQUE
}
ERROR  ::=  CLASS
{
        &ParameterType OPTIONAL,
        &code             INTEGER UNIQUE
}
```

Exempel på definition av en operation med användning av ovan definierad OPERATION klassen:


```
invertMatrix  OPERATION  ::=
{
        &ArgumentType Matrix,
        &ResultType       Matrix,
        &Errors           {determinantIsZero}
        &operationCode 7
}
determinantIsZero  ERROR  ::=
{
        &errorCode        1
}
```

Man kan också definiera en egen syntax för en ny klass. Med en sådan syntax-definition kan ovanstående exempel t.ex. se ut så här:

2b-137

```
invertMatrix  OPERATION  ::=
{
        ARGUMENT        Matrix,
        RESULT          Matrix,
        ERRORS          {determinantIsZero}
        CODE    7
}
determinantIsZero  ERROR  ::=
{
        CODE            1
}
```


Den syntax-definition, som gav exemplet ovan, ser ut så här:

2b-138

```
OPERATION  ::=  CLASS
{
        &ArgumentType OPTIONAL,
        &ResultType     OPTIONAL,
        &Errors         ERROR OPTIONAL,
        &Linked         OPERATION OPTIONAL,
        &resultReturned BOOLEAN DEFAULT TRUE,
        &code           INTEGER UNIQUE
}
WITH SYNTAX
{
        [ARGUMENT       &ArgumentType]
        [RESULT         &ResultType]
```

2b-139

```
        [RETURN RESULT  &resultReturned] (forts på nästa sida)
        [ERRORS         &Errors]
        [LINKED         &Linked]
        CODE            &operationCode
}
ERROR  ::=  CLASS
{
        &ParameterType OPTIONAL,
        &errorCode      INTEGER UNIQUE
}
WITH SYNTAX
{
        [PARAMETER      &ParameterType]
        CODE            &errorCode
}
```

ASN.1/1994 part 3: Constraint specification

2b-140

Constraint och *subclass* är två ord för ungefär samma sak. Constraint i form av en kommentar infördes i ASN.1 1994:

Definition:

```
ENCRYPTED { ToBeEnciphered }  ::=  BIT STRING
( CONSTRAINED-BY
{
- - must be the result of the encipherment of some
- - BER-encoded value of - - ToBeEnciphered
} }
```

Use:

```
ENCRYPTED { SecurityParameters }
```

ASN.1/1994 part 4: Parameterisation

Definition:

```
SIGNED { ToBeSigned } ::= SEQUENCE
{
        authenticated-data      ToBeSigned,
        autheticator            BIT STRING
}
```

Use:

```
SIGNED { OrderInformation }
```

Vilket då är samma sak som om man skrivit:

```
SEQUENCE
{
        authenticated-data      OrderInformation,
        authenticator           BIT STRING
}
```


ASN.1/1994 part 4: Parameterisation


```
OPTIONALLY-SIGNED { ToBeSigned } ::= CHOICE
{
        unsigned-data   [0]     ToBeSigned,
        signed-data     [1]     SIGNED { ToBeSigned }
}
```



```
From: Marshall T. Rose <mrose@dbc.mtview.ca.us>
Date: 12 jul 1995 05:12
... ...

Combining ASN.1 and high-performance is oxymornonic.

ASN.1 is probably the greatest failure of the OSI
effort, it led hundreds of engineers, including
myself, to devise data structures that were far too
complicated for their own good.
```

(Oxymoron = Self-contradiction)

(Marshall T. Rose is a well-known previous OSI expert who has turned into one of the most vocal OSI enemies.)


```
From: Colin Robbins <c.robbins@nexor.co.uk>
Date 13 Jul 1995 16:58

Let me see if I have understood this debate.
X.400 is a brontosarus, because it uses ASN.1.
SMTP is a monkey because it does not.

Where does that leave the SNMPv2 Protocol, desgined
by the Internet community, co-auther one Marshall T.
Rose.  It uses ASN.1. I thought leopards didn't
change their spots!

There are plenty or reasons to knock X.400, but the
use of ASN.1 is not one of them.  Sure it has its
faults, but BOTH the Internet and OSI communities
are using it.
```

Litteratur för den somn vill lära sig mera

Douglas Steedman: Abstract Syntax Notation One ASN.1 The tutorial & Reference. Technology Appraisals 1990. *(Kan köpas i bokhandeln, boken är mycket dyr.)*

X.208 (ASN.1)

X.209 (BER)

X.219 (ROS)

X.420 (Interpersonal Messaging Service)

X.500 (Directory System)

*:96 Overheads

Part 2c: URL, Media types

More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 02-03-21 19.17

URL, Uniform Resource Locator

An URL identifies a resource, such as a document, as stored in one particular location, and an access protocol to connect to the resource or, in the case of a document, to retrieve it.

References:

RFC 1738: Uniform Resource Locators (URL), by T. Berners-Lee, L. Masinter and M. McCahill. URL
`ftp://ftp.sunet.se/pub/Internet-documents/rfc/rfc1738.txt`

RFC 1808: Relative Uniform Resource Locators, by R. Fielding, URL
`ftp://ftp.sunet.se/pub/Internet-documents/rfc/rfc1808.txt`

Examples:

`http://dsv.su.se/~jpalme`	identifies my personal home page, as retrieved with the HTTP (WWW) protocol.
`ftp://ftp.sunet.se/pub/Internet-documents/rfc/rfc1738.txt`	identifies the copy of RFC1738 stored at FTP.SUNET.SE for retrieval using FTP.
`http://ftp.sunet.se/pub/Internet-documents/rfc/rfc1738.txt`	identifies the copy of RFC1738 stored at FTP.SUNET.SE for retrieval using HTTP.

URL schemes standardized in RFC 1738

`ftp`	File Transfer protocol
`http`	Hypertext Transfer Protocol
`gopher`	The Gopher protocol
`mailto`	Electronic mail address
`news`	USENET news
`nntp`	USENET news using NNTP access
`telnet`	Reference to interactive sessions
`wais`	Wide Area Information Servers
`file`	Host-specific file names
`prospero`	Prospero Directory Service

Character set in URLs (not in referenced document)

Only US-ASCII allowed

Unsafe characters: `space < > " # % { } | \ ^ ~ [ ] ``

Reserved characters in some URL schemes: `; / ? : @ = &`

Unsafe characters must be encoded in transport.

Reserved characters not used in their reserved meaning must be encoded or may not be used.

Safe characters is the rest of US-ASCII, i.e. `A-Z, a-z, 0-9, $ - _ + ! * ' ( ) ,`

Encoding of unsafe characters in URL-s

In addition, octets may be encoded by a character triplet consisting of the character "%" followed by the two hexadecimal digits (from "0123456789ABCDEF") which forming the hexadecimal value of the octet. (The characters "abcdef" may also be used in hexadecimal encodings.)

Examples:

The string
`"Donald Duck"`

is encoded as
`%22Donald%20Duck%22`

Top-level URL Syntax:

```
<scheme>:<scheme-specific-part>
```

Common Internet Scheme Syntax

```
//<user>:<password>@<host>:<port>/<url-path>
```

Examples of three URL-s referring to the same document

```
ftp://ftp.dsv.su.se/users/Jacob.Palme/draft-ietf-mailext-new-fields-05.txt
ftp://anonymous:@ftp.dsv.su.se/users/Jacob.Palme/draft-ietf-mhtml-info-01.txt
ftp://anonymous:@ftp.dsv.su.se:21/users/Jacob.Palme/draft-ietf-mhtml-info-01.txt
```

Examples of five URL-s referring to the same directory

```
ftp://jpalme:password@ester.dsv.su.se
ftp://jpalme:password@ester.dsv.su.se/home0/ester/dsv/dsv-jp
ftp://jpalme:password@ester.dsv.su.se:21/home0/ester/dsv/dsv-jp
ftp://jpalme:password@ester.dsv.su.se:21/home0/ester/dsv/dsv-jp/
ftp://jpalme:password@ester.dsv.su.se:21
/home0/ester/dsv/dsv-jp/;type=d
```


Relative URLs

Based on this Base URL:

```
URL:http://a/b/c/d;p?q#f
```

The following URLs are resolved as shown:

URL	Resolved URL
g:h	g:h
g	http://a/b/c/g
./g	http://a/b/c/g
g/	http://a/b/c/g/
/g	http://a/g
//g	http://g
?y	http://a/b/c/d;p?y
g?y	http://a/b/c/g?y
g?y/./x	http://a/b/c/g?y/./x
#s	http://a/b/c/d;p?q#s
g#s	http://a/b/c/g#s
g#s/./x	http://a/b/c/g#s/./x

URL	Resolved URL
g?y#s	http://a/b/c/g?y#s
;x	http://a/b/c/d;x
g;x	http://a/b/c/g;x
g;x?y#s	http://a/b/c/g;x?y#s
.	http://a/b/c/
./	http://a/b/c/
..	http://a/b/
../	http://a/b/
../g	http://a/b/g
../..	http://a/
../../	http://a/
../../g	http://a/g

HTTP URL syntax

```
http://<host>:<port>/<path>?<searchpart>
```

Example of an HTTP Query URL

A search for "Donald Duck" to Alta Vista is encoded as:

```
http://altavista.digital.com/cgi-bin/query?pg=q&what=web&fmt=.&q=%22Donald+Duck%22
```

Reference to fragments of an HTML document

Relative reference:

```
#anchor1003017
```

Absolute reference:

```
http://www.dsv.su.se/~jpalme/ietf/jp-ietf-home#anchor1003017
```

Markup in the referenced HTML document

```
<A NAME="anchor1003017"></A><BR>
```

Part of the URL?

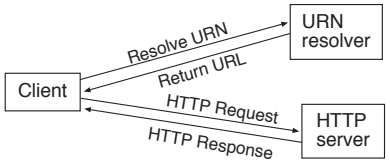
Section preceded by ? are regarded as part of the URL itself, but section preceded by # are not regarded as part of the URL itself.

URL, URI, URN, URC

URI = Uniform Resource Identifier

URL = Uniform Resource Locator

An URI, which refers to a particular copy of a document stored on a particular host. May have to be changed if the document is moved. Belongs to registered URI/URL schemes.



URN = Uniform Resource Name

A URI scheme for various namespaces. Refers to a document, wherever it is stored. Can be resolved into an URL by an URN resolver. An URN resolver may locate the copy of a mirrored document which is clostest to the requestor. Begins with “urn:” followed by the namespace, followed by the value, for example “urn:isbn:

URI = URL or URN

URC = Uniform Resource Characteristics

The purpose or function of a URC is to provide a vehicle or structure for the representation of URIs and their associated meta-information.

Media types

Initially defined in RFC 1521. New media types can be registered with IANA, usually an RFC defining the type is provided. Registered media types are listed in `ftp://ftp.isi.edu/in-notes/iana/assignments/media-types`.

Format:

```
<type> "/" <subtype> [ ";" *(<para> "=" <value>) ]
```

Primary media types

Type	Description
text	Mainly text. Can always have "charset" parameter. Default for charset is sometimes US-ASCII.
multipart	Consists of several parts, which each may be of different type.
message	An encapsulated message (usually includes message heading).
application	Executable code. Note that postscript is application/postscript, not text/postscript.
image	Still picture.
audio	Sound.
video	Moving picture (may include sound).

Some important subtypes:

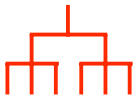
The *text* media type

Mainly text. Can always have "charset" parameter. Default for charset is US-ASCII in e-mail, ISO 8859-1 in HTTP 1.0.

Type	Description
text/plain	Text without other formatting codes than horizontal tab, CRLF and form feed.
text/richtext, text/enriched	Two simpler formatting schemes than HTML and SGML.
text/html	Hypertext Markup Language, The main document format in the WWW. Version 2.0 is defined in RFC 1866. HTML is an application of SGML.
text/sgml	Standard Generalized Markup Language. ISO standard.
text/rfc822-headers	Headers from a mail message (returned in a delivery status notification).

HTML (.HTML, .HTM) = mjuk formattering,
Postscript (.PS) och Adobe Acrobat (.PDF) = hård formattering

The *multipart* media type



Contains several parts, which may be of different types

Type	Description
multipart/mixed	A sequence of parts to be displayed in sequence.
multipart/alternative	Several versions of the same information, simplest first, recipient displays the most advanced version it can handle. Example: plain text versus richtext versus html.
multipart/digest	A set of messages, headings sometimes abbreviated.
multipart/parallel	Parts to be shown at the same time.

	Example: Image and sound.
multipart/related	Related parts, such as an HTML document and inline images, RFC 1872.
multipart/report	Delivery status and other notifications RFC 1892
multipart/form-data	Using HTML forms to upload files from the client, RFC 1867.
multipart/header-set	Set of data, some of which is system-specific and some of which is in MIME standard types
multipart/appledouble	Binary Macintosh files
multipart/voice-message	Using Internet mail for communication between voice-mail machines, RFC 1911

Content-Disposition

Content-Disposition: ("Inline" / "Attachment")

The *message* media type

Usually contains a message

Type	Description
message/rfc822	Internet e-mail message.
message/partial	One of a set of messages which are to be combined.
message/external-body	Message, whose body is referenced and not included. Access types: FTP, ANON-FTP, TFTP, AFS, LOCAL-FILE, MAIL-SERVER, Content-ID, URL.
message/news	A usenet news article.
message/http	A document which has been transmitted through HTTP.
message/delivery-status	Delivery status report.

The *application* media type (not complete)

Type	Description
octet-stream	Any binary data.
postscript	Adobe postscript page description language.
rtf	Rich Text Format, Microsoft standard for exchange of documents between word processing software, also supported by other vendors than Microsoft.
pdf	Adobe Acrobat.
activemessage	How to connect to an Active Mail application at a remote host.
mac-binhex40	Macintosh binary-to-text conversion method
remote-printing	RFC 1486: Printing in a remote location
msword, cybercash,	Vendor-specific formats.

wordperfect5.1, vnd-framemaker, etc.

The *audio* media type

Contains sound.

Type	Description
audio/basic, audio/32kadpcm	Two different sound encoding methods

The *video* media type

Contains moving pictures, can include sound.

Type	Description
mpeg, quicktime, vnd.vivo	Two different video encoding methods

The charset attribute

US-ASCII	Plain US-ASCII, not other ISO 646 variants. 7 bits. Default in e-mail.
ISO-8859-1	Also known as ISO latin 1. 8 bits. Default in WWW.
ISO-8859-?	Other variants of ISO 8859 for different language groups.
UTF-8	Unicode/ISAO 10646 with the UTF-8 encoding

*:96 Overheads

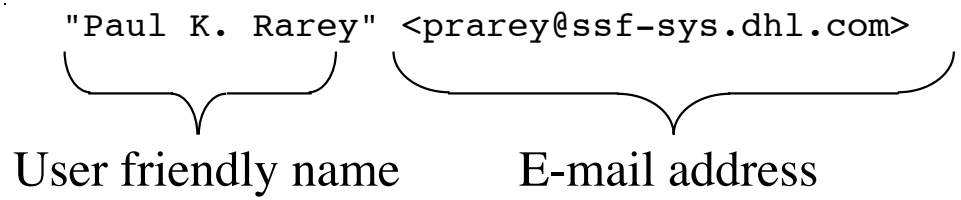
Part 3a: E-mail introduction

More about this course about Internet application protocols can be found at URL:

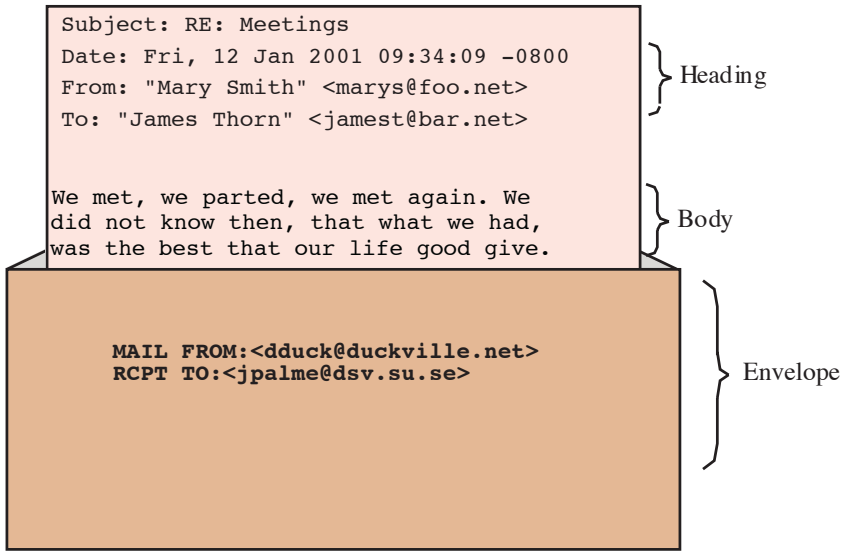
`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 23 Dec 2005

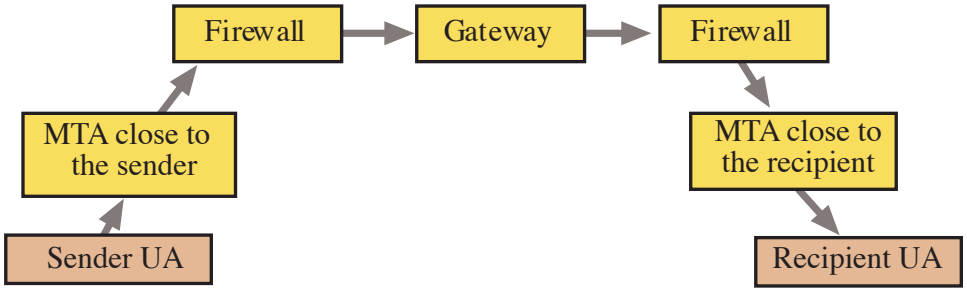
Internet e-mail address format



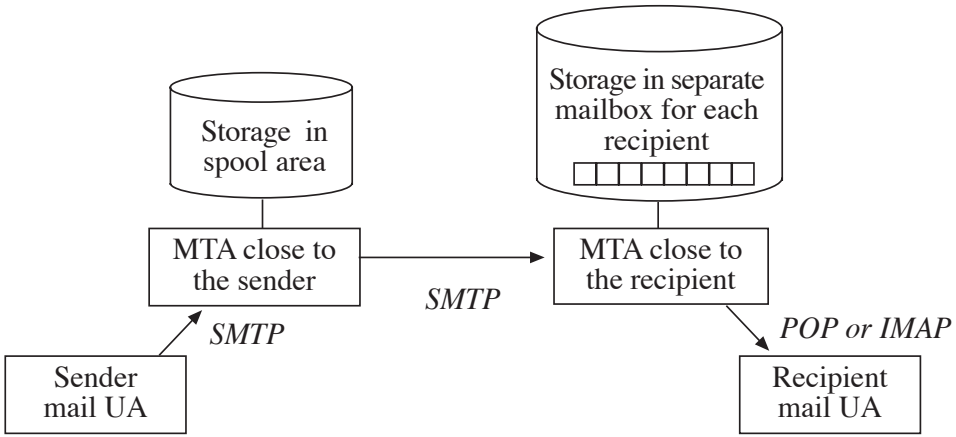
Envelope, Content, Heading and Body



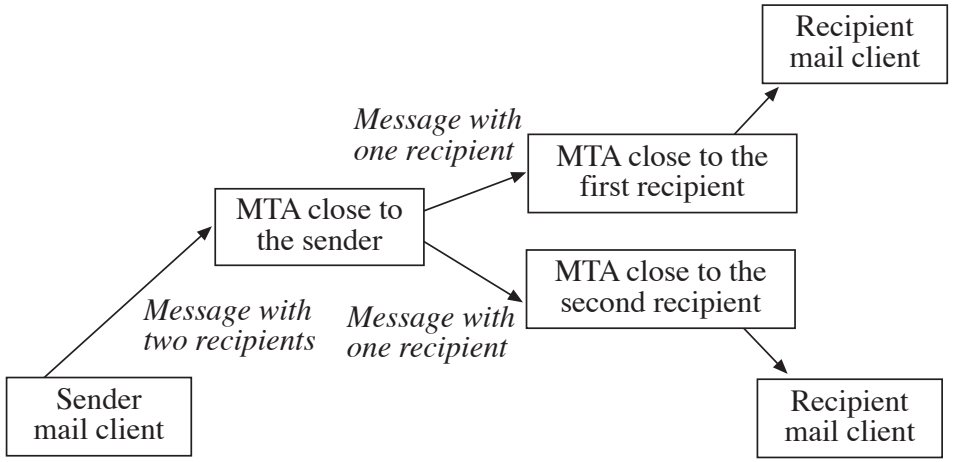
Multiple Relaying because of Firewalls



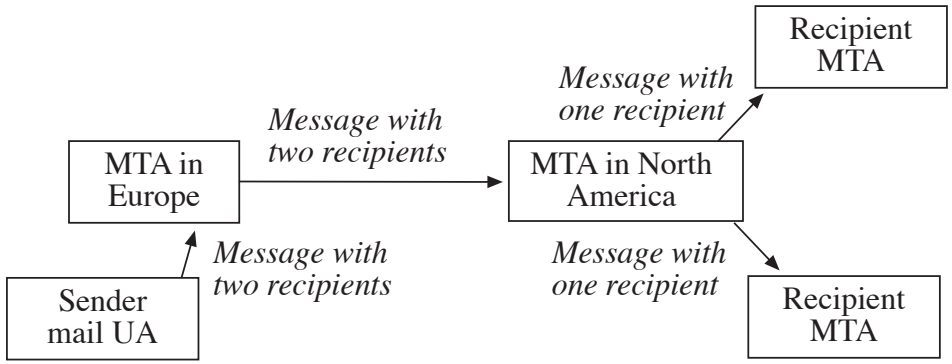
Storage of Mail in MTAs



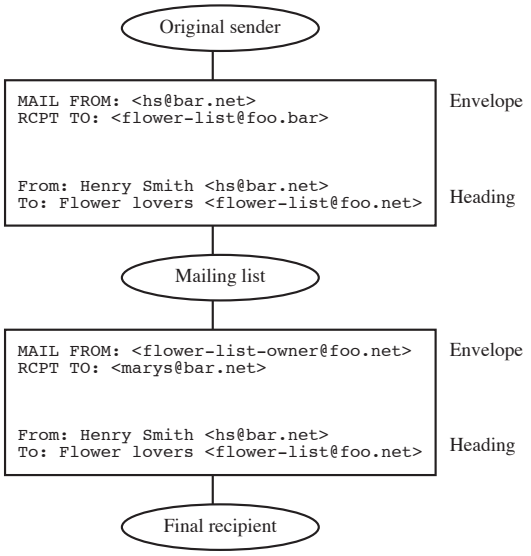
Split of Messages in MTAs



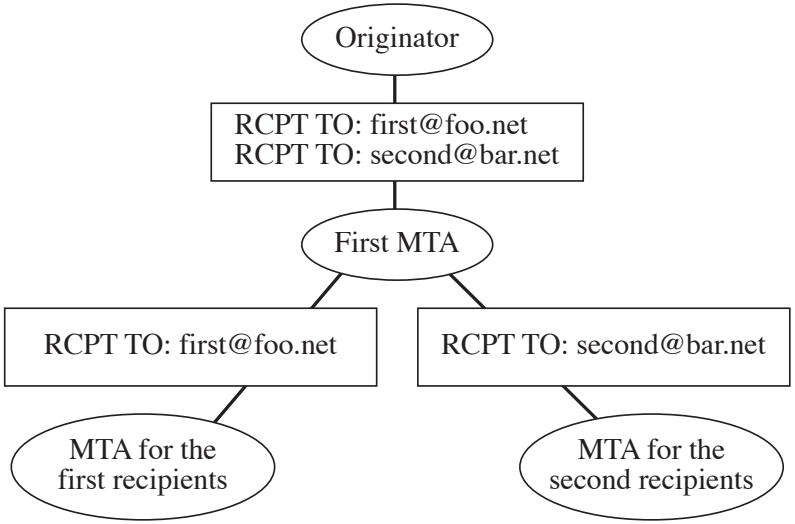
Saving Netload by Split in MTAs



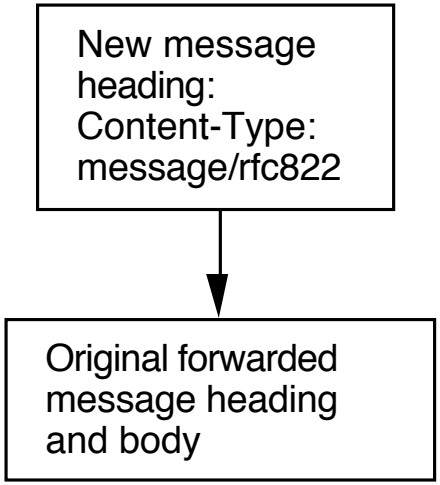
From and MAIL FROM for Mailing Lists



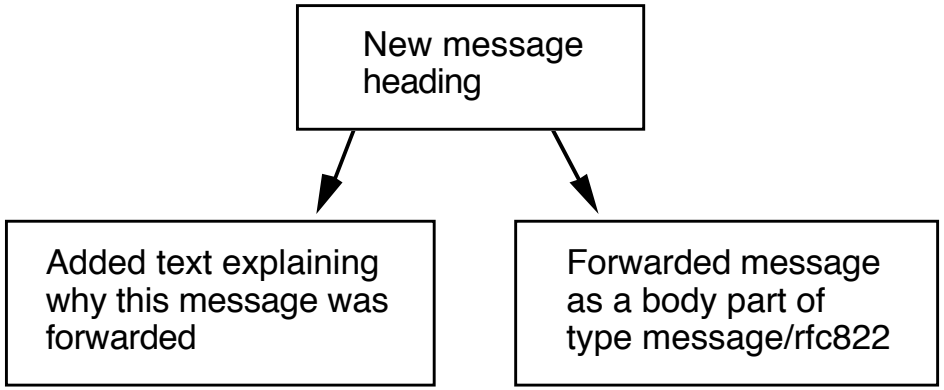
RCPT TO versus To after Split



Forwarding with a MIME Message/rfc822



Forwarding with a MIME multipart message

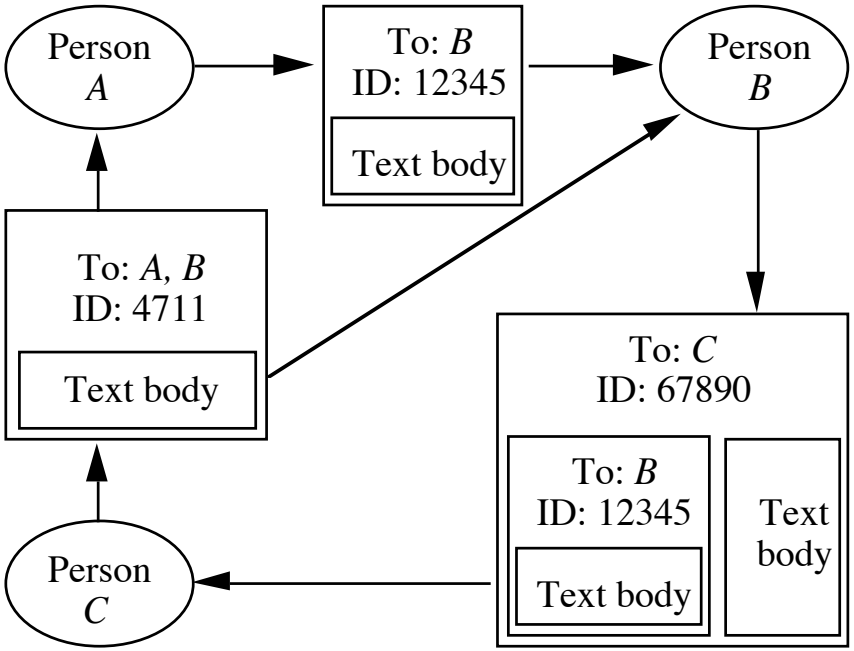


Resending and Forwarding

Original message	Date: 14 Jan 2005 09:35 From: Donald Duck <dduck@northpole.foo> To: Jacob Palme <jpalme@dsv.su.se>
Resent message	Date: 14 Jan 2005 09:35 Resent-Date 14 Jan 2005 10:35 From: Donald Duck <dduck@northpole.foo> Resent-From: Jacob Palme <jpalme@dsv.su.se> To: Jacob Palme <jpalme@dsv.su.se> Resent-To: Ducklovers <dlovers@dsv.su.se>
Forwarded message	Date: 14 Jan 2005 10:35 From: Jacob Palme <jpalme@dsv.su.se> To: Ducklovers <dlovers@dsv.su.se> > Header and body of original message

Original message	Date: 14 Jan 2005 09:35 From: Donald Duck <dduck@northpole.foo> To: Jacob Palme <jpalme@dsv.su.se>
Mime forwarding variant 1	Date: 14 Jan 2005 10:35 From: Jacob Palme <jpalme@dsv.su.se> To: Ducklovers <dlovers@dsv.su.se> Content-Type: message/rfc822 Date: 14 Jan 2005 09:35 From: Donald Duck <dduck@northpole.foo> To: Jacob Palme <jpalme@dsv.su.se>

Original message	Date: 14 Jan 2005 09:35 From: Donald Duck <dduck@northpole.foo> To: Jacob Palme <jpalme@dsv.su.se>
Mime forwarding variant 1	Date: 14 Jan 2005 10:35 From: Jacob Palme <jpalme@dsv.su.se> To: Ducklovers <dlovers@dsv.su.se> Content-Type: Multipart/mixed Content-Type: message/rfc822 Date: 14 Jan 2005 09:35 From: Donald Duck <dduck@northpole.foo> To: Jacob Palme <jpalme@dsv.su.se> Content-Type: Text/plain Comment on forwarded message



Methods of e-mail forwarding

(a) Add new Resent-headers to the original message.

(b) The forwarded message is made into a body part of type message/rfc822 in a new multipart message:

(c) The text of the forwarded message is copied into the text of the new message with copy marks.

Which method is best if the forwarded message had a digital seal?

MAIL FROM (Envelope) versus From (Header)

Original message	RCPT TO:<cmclist@host.net> MAIL FROM:<jpalme@dsv.su.se> Date: 14 Jan 2005 09:35 From: Jacob Palme <jpalme@dsv.su.se> To: CMC mailing list <cmclist@host.net>
Forwarded message	RCPT TO:<mary@host.net> MAIL FROM:<cmclist-owner@host.net> Date: 14 Jan 2005 09:35 From: Jacob Palme <jpalme@dsv.su.se> To: CMC mailing list <cmclist@host.net>

*:96 Overheads

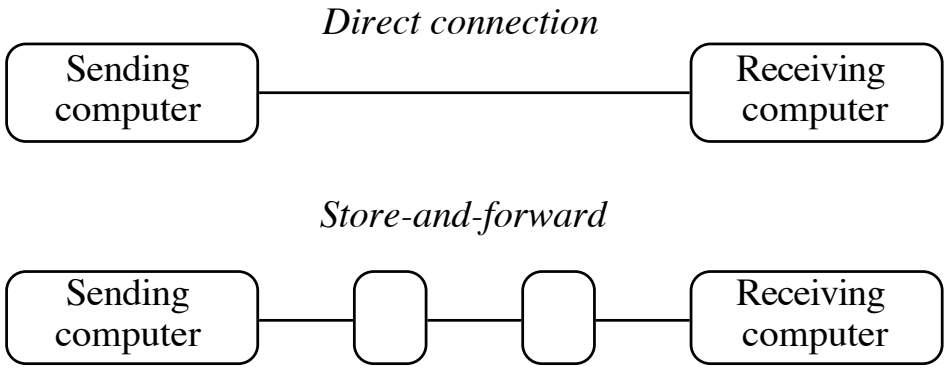
Part 3b: E-mail basics

More about this course about Internet application protocols can be found at URL:

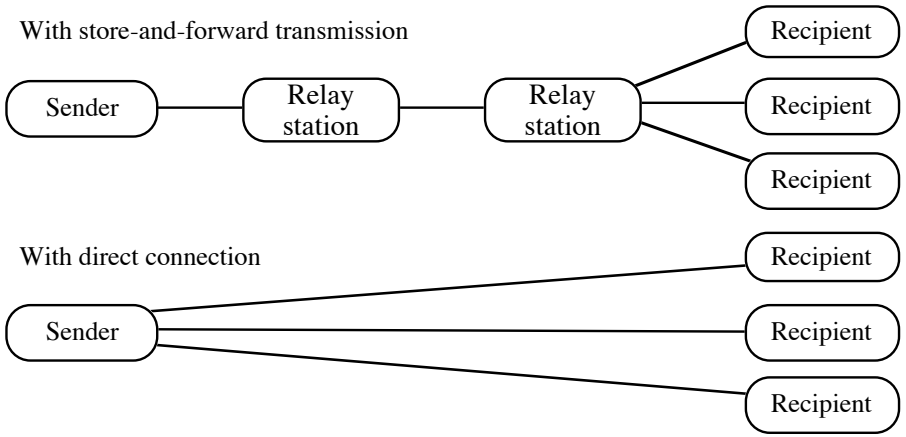
`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 23 Dec 2005

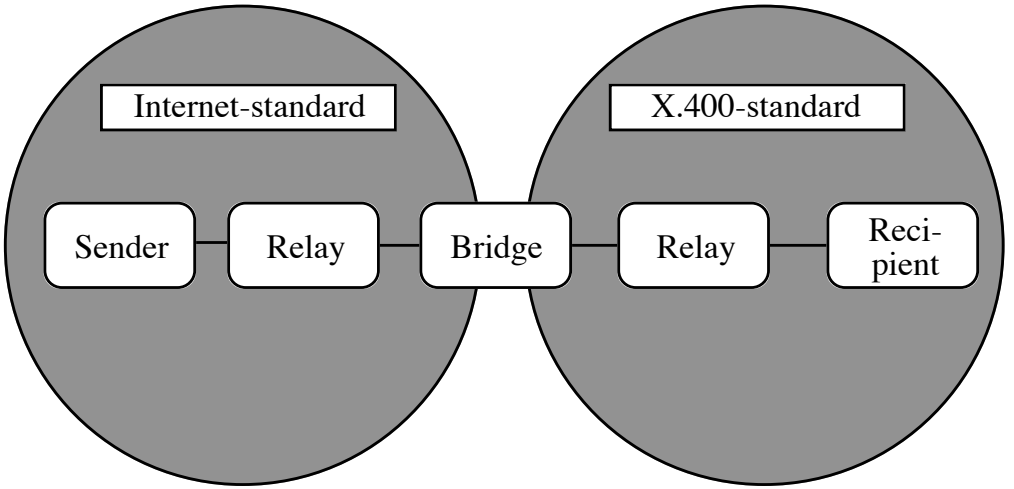
Direct connection and store-and-forward



Many distant recipients



Gateways' use of store-and-forward



Store-and-forward pros and cons

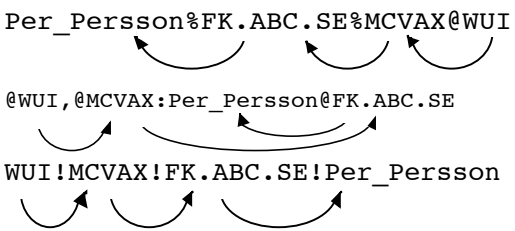
+ Distribution of tasks between specialized servers. But direct transmission can employ special routing information servers.
+ Reduced cost for message to many distant recipients.
+ Gateways usually store-and-forward-based.
– Reliability
– Can be more expensive because relayers must be paid.

Spooling - a limited kind of store-and-forward

– No direct and immediate confirmation that the message has been delivered.
+ The sender need not wait during the transmission.
+ Temporary connection problems hidden from the user.

Absolute and relative addresses

An *absolute address* is the same address for a certain recipient, irrespective of where the message is sent from. A *relative address* indicates one or more relay stations on the route to the recipients.



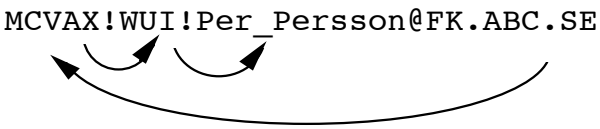
Grey book mail format

RFC 822 format

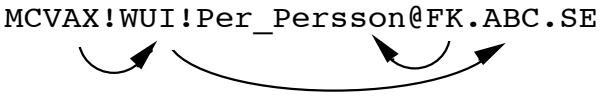
UUCP format

Mixed relative addressing

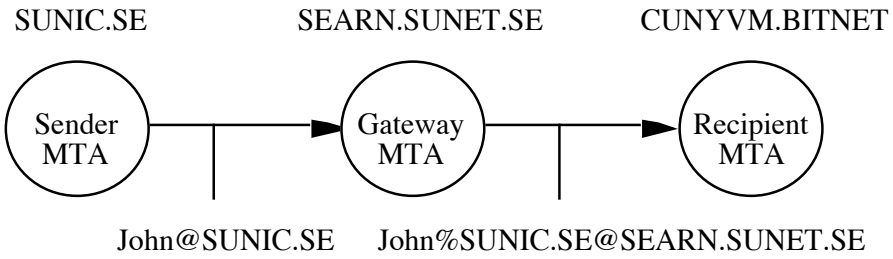
RFC 822 interpretation



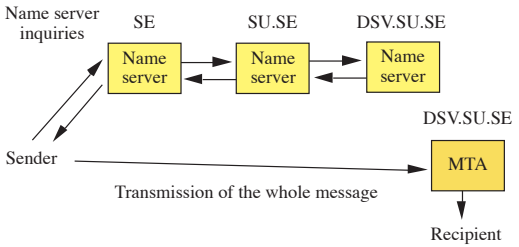
older UUCP interpretation



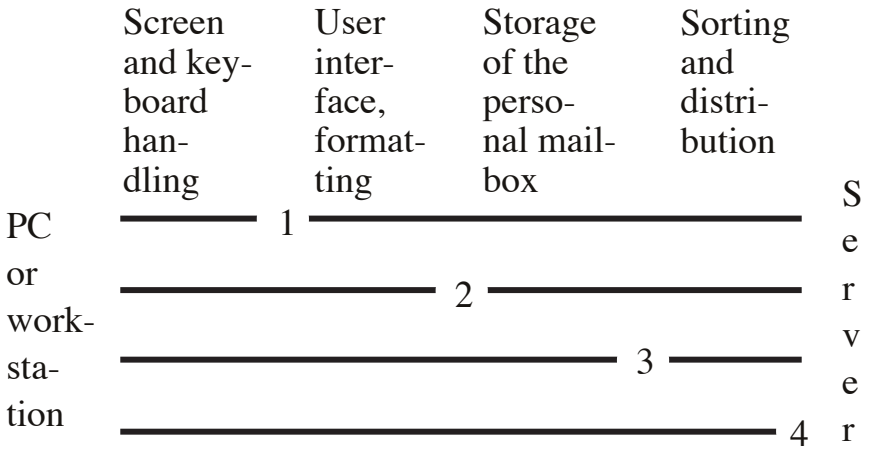
Why gateways produce relative addresses



Use of DNS servers for routing



PC-Server E-mail Architectures



Protocols: POP (3), IMAP (2, 3)

Public/secret key encryption

encrypted text = f_1(original text)

original text = f_2(encrypted text)

Can f_2 be derived from f_1?

Pros and cons of public key encryption

\+ Solves partly key transportation problem

– More CPU-time consuming

Authentication, authorization

- To verify the sender of a message
- Payments, agreements
- UA-UA or MTA-MTA

Authentication methods

(a) Passwords
(b) Specially designed networks
(c) Public key cryptography

Three levels of protection of message transmission:

(1) The agents identify each other using noninvertible forms of ordinary passwords. This is called *weak authentication*.
(2) The agents identify each other using public key encryption algorithms. This is called *strong authentication*.

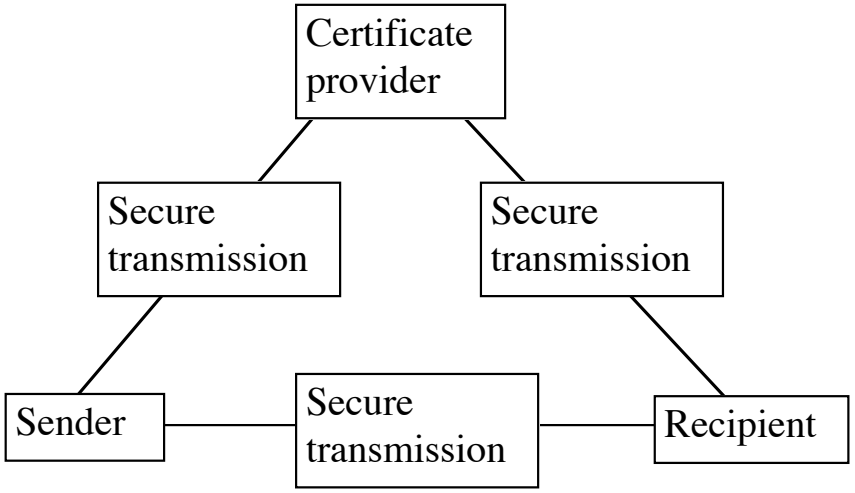
(3) Strong authentication is combined with encryption of all messages during the whole transmission.

Digital Signatures and Digital Seals

Methods: Secret key encryption of signature or checksum, which anyone can decrypt with public key

- Number of interactions
- Need of a neutral third party
- Bilateral or open to groups

Certificate Authorities



*:96 Overheads

Part 3c: E-mail, SMTP, RFC822, MIME

More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 23 Dec 2005

SMTP
- an interactive protocol

(Defined in RFC 821 and RFC2821)

HELO/EHLO	Opening session
MAIL FROM	Identifying sender
RCPT TO	Identifying recipient
DATA	Sending message body
VRFY	Check recipient name
EXPN	Expand alias/mailing list

SMTP delivery modes

MAIL FROM	To the recipient's mailbox
SEND FROM	Display on logged-in terminal
~~SOML FROM~~	Display on terminal if logged in, otherwise deliver to mailbox
~~SAML FROM~~	Deliver to mailbox and optionally display on terminal if logged in

Example of an SMTP dialogue

Sending agent command	*Responding agent*
`HELO dsv.su.se`	`250 nexor.co.uk`
`MAIL FROM: jpalme@dsv.su.se`	`250 OK`
`RCPT TO: <j.onions@nexor.co.uk>`	`250 OK`
`RCPT TO: <seb@nexor.co.uk>`	`250 OK`
`DATA`	`354 Start mail input; end with <CRLF>.<CRLF>`
`... the lines of text ...`	
`.`	`250 OK`
`QUIT`	`221 nexor.co.uk`

SMTP syntax - 1

```
commands  ::=     "HELO "    domain   CRLF
          /  "MAIL FROM: " route-addr CRLF
          /  "MAIL FROM: " "<>" CRLF
          /  "RCPT TO: "   route-addr CRLF
          /  "DATA"  CRLF
          /  "NOOP"  CRLF
          /  "RSET"  CRLF
          /  "QUIT"  CRLF
          /  optional

optional ::= "SEND FROM: " route-addr CRLF
          /  "SOML FROM: " route-addr CRLF
          /  "SAML FROM: " route-addr CRLF
          /  "VRFY "       string  CRLF
          /  "EXPN "       string  CRLF
          /  "TURN"  CRLF
          /  "HELP"   [" " string] CRLF
```


SMTP syntax - 2

```
response ::=     *(code ["-"] [" " *text] CRLF)
                 code      [" " *text] CRLF

code::=     ("1" / "2" / "3" / "4" / "5")
            ("0" / "1" / "2" /  / "5")
            1DIGIT

string ::=  1*(stext / quoted-pair)

stext  ::=  <any character, not including space or
specials>

text   ::=  <any character, including bare CR and
bare LF, but not including CRLF>
```


SMTP Reply codes (RFC 821)

First digit:

1 Positive preliminary (not used in SMTP)

2 Success

3 Ready but requires additional info

4 Transient failure

5 Permanent negative

Second digit:

0 Syntax (error)

1 Information requested in reply

2 Transport service issue

5 Application-specific issue

Third Digit

To distinguish between reply codes with the same first two digits

Examples of reply codes to MAIL FROM:

250 Originator accepted

452 Out of local storage

500 Command syntax error

TRACE list for a spam message:

```
Return-Path: administrator@unixstory.org
X-Original-To: jpalme@dsv.su.se
Delivered-To: jpalme@dsv.su.se
```

```
Received: from av-in2.su.se (av-in2.su.se
[130.237.93.211])
 by unni.dsv.su.se (Postfix) with ESMTP id
246698B3B5
 for <jpalme@dsv.su.se>; Tue, 22 Nov 2005 12:16:46
+0100 (CET)
```

```
Received: from localhost (av-in2.su.se [127.0.0.1])
 by av-in2.su.se (Postfix) with ESMTP id
167CA23EEBE
 for <jpalme@dsv.su.se>; Tue, 22 Nov 2005 12:16:46
+0100 (CET)
```

```
Received: from av-in2.su.se ([127.0.0.1])
 by localhost (av-in2.su.se [127.0.0.1]) (amavisd-
new, port 10024) with LMTP
 id 16826-01-41 for <jpalme@dsv.su.se>; Tue, 22 Nov
2005 12:16:45 +0100 (CET)
```

```
Received: from mx1.su.se (mx1.su.se
[130.237.162.110])
 by av-in2.su.se (Postfix) with ESMTP id
6B4F223EE97
 for <jpalme@dsv.su.se>; Tue, 22 Nov 2005 12:16:45
+0100 (CET)
```

```
Received: from mta3.iomartmail.com
(mta3.iomartmail.com [62.128.193.153])
 by mx1.su.se (Postfix) with ESMTP id 2EF382400A
 for <jpalme@dsv.su.se>; Tue, 22 Nov 2005 12:16:44
+0100 (CET)
```



```
Received: from mta3.iomartmail.com (localhost
[127.0.0.1])
 by mta3.iomartmail.com (8.12.8/8.12.8) with ESMTP
id jAMBEkBw004968;
 Tue, 22 Nov 2005 11:14:46 GMT
```

```
Received: from unkown ([196.202.65.92])
 (authenticated bits=0)
 by mta3.iomartmail.com (8.12.8/8.12.8) with ESMTP
id jAMBDjIa003129;
 Tue, 22 Nov 2005 11:14:41 GMT
```

```
Message-Id:
<200511221114.jAMBDjIa003129@mta3.iomartmail.com>
From: "Paypal Billing Departmentemail"
<administrator@unixstory.org>
Subject: Your Account Will Be Suspended
To: p1@canada.com
Content-Type: text/html;iso-8859-1
Reply-To: administrator@unixstory.org
```



```
Date: Tue, 22 Nov 2005 13:14:37 +0200
X-Priority: 3
X-Library: Indy 8.0.25
X-Virus-Scanned: by amavisd-new at av-in.su.se
```

```
X-Spam-Status: No, score=4.6 tagged_above=-99
required=7
 tests=[BAYES_50=0.001, HTML_MESSAGE=0.001,
MIME_HEADER_CTYPE_ONLY=0,
 MIME_HTML_ONLY=0.001, RCVD_IN_BL_SPAMCOP_NET=1.558,
 SARE_HEAD_XLIB_INDY2=0.639, X_LIBRARY=2.4]
X-Spam-Score: 4.6
X-Spam-Level: ****
```

```
X-Spamfire-UID:
<200511221114.jAMBDjIa003129@mta3.iomartmail.com>
```


ESMTP (RFC 1869 SMTP Service Extensions; Obsoletes RFC 1651)

EHLO command

A client supporting ESTMP should open SMTP communication with EHLO instead of HELO. The full syntax of the EHLO command is

```
ehlo-cmd ::= "EHLO" SP domain CR LF
```

Responses from the server to the EHLO command:

Successful	250 domain [SP greeting] CR LF 250- domain [SP greeting] CR LF *(250- ehlo-line CR LF) 250 SP ehlo-line CR LF ehlo-line ::= ehlo-keyword * (SP ehlo-param)
Failure, try HELO or QUIT	554 Cannot list service extensions
Command recognized but argument unacceptable	501
EHLO recognized but not implemented	502
SMTP service temporaily not available	421

SMTP service extensions

Service extension	Keyword	Parameters	Verb	Defined in RFC
Send	SEND	none	SEND	2821, 1869
Send or Mail	SOML	none	SOML	821, 1869
Send and Mail	SAML	none	SAML	821, 1869
Expand	EXPN	none	EXPN	2821, 1869
Help	HELP	none	HELP	2821, 1869
Turn	TURN	none	TURN	821, 1869
Pipelining	PIPELINING	none	none	1854
Message size declaration	SIZE	adds optional parameter size-value ::= 1*20DIGIT no of octets	none	1870

Checkpoint/ restart	CHECKPOINT	adds optional parameter TRANSID to MAIL FROM command	none	1845
Large and binary MIME messages	CHUNKING	BDAT is used instead of DATA, and takes as parameter packet length and last packet indication	BDAT	1830
8bit-MIMEtransport	8BITMIME	adds optional parameter BODY to MAIL FROM, values 7BIT and 8BITMIME	none	1652
Delivery Status Notification Extension	DSN	adds optional parameters NOTIFY and ORCPT to RCPT command and RET and ENVID to the MAIL command	none	1891, 1892, 1894

Examples of ESTMP establishment interactions

(1)
Only mandatory SMTP commands provided

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 dbc.mtview.ca.us SMTP service ready
C: EHLO ymir.claremont.edu
S: 250 dbc.mtview.ca.us says hello
```

---

(2)

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 dbc.mtview.ca.us SMTP service ready
C: EHLO ymir.claremont.edu
S: 250-dbc.mtview.ca.us says hello
```

Basic optional services: EXPN, HELP;

```
S: 250-EXPN
S: 250-HELP
```

Standard service extension: 8BITMIME;

```
S: 250-8BITMIME
```

Unregistered services: XONE and XVBR

```
S: 250-XONE
S: 250 XVRB
```

---

(3)
ESTMP not supported

```
S: wait for connection on TCP port 25>
C: <open connection to server>
S: 220 dbc.mtview.ca.us SMTP service ready
C: EHLO ymir.claremont.edu
S: 500 Command not recognized: EHLO
C: HELO ymir.claremont.edu
```


SMTP command pipelining (RFC 1854)

Without pipelining: 9 turnarounds:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP ready
C: HELO dbc.mtview.ca.us
S: 250 innosoft.com
C: MAIL FROM:<mrose@dbc..ca.us>
S: 250 sender <mrose@dbc..ca.us> OK
C: RCPT TO:<ned@innosoft.com>
S: 250 recipient <ned@in.com> OK
C: RCPT TO:<dan@in.com>
S: 250 recipient <dan@in.com> OK
C: RCPT TO:<kvc@in.com>
S: 250 recipient <kvc@in.com> OK
C: DATA
S: 354 enter mail, end with line
containing only "."
 ...
C: .
S: 250 message sent
C: QUIT
S: 221 goodbye
```

With pipelining: 4 turnarounds:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP ready
C: EHLO dbc.mtview.ca.us
S: 250-innosoft.com
S: 250 PIPELINING
C: MAIL FROM:<mrose@dbc..ca.us>
C: RCPT TO:<ned@innosoft.com>
C: RCPT TO:<dan@innosoft.com>
C: RCPT TO:<kvc@innosoft.com>
C: DATA
S: 250 sender <mrose@dbc..ca.us> OK
S: 250 recipient <ned@in.com> OK
S: 250 recipient <dan@in.com> OK
S: 250 recipient <kvc@in.com> OK
S: 354 enter mail, end with line
 containing only "."
 ...
C: .
C: QUIT
S: 250 message sent
S: 221 goodbye
```

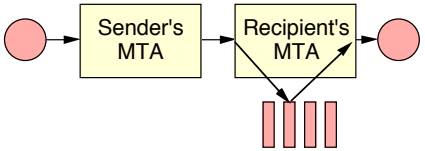
Delivery status notifications

RFC 1891: SMTP service extension

RFC 1892: Multipart/report content-type

RFC 1893: Enhanced status codes

RFC 1894: Delivery Status Notification format



SMTP command	Optional parameter	Description
RCPT TO	NOTIFY	Values: NEVER SUCCESS FAILURE DELAY (= willingness to accept notifications if delivery is delayed)
RCPT TO	ORCPT	Original sender-specified recipient address (needed to allow recpient of notifications to correlate notifications with original recipients)
MAIL FROM	RET	Values: FULL HDRS
MAIL FROM	ENVID	Transaction ID to be returned with notification

The Multipart/report MIME Content Type (RFC 1892)

Part 1 (mandatory):	Human readable message	Any content type. Multipart/alternative can be used, for example to give message in more than one language.
Part 2 (mandatory):	Machine parsable account of reported event	Message/delivery-status (defined in RFC 1894)
Part 3 (optional):	Returned message or portion thereof	Text/rfc822-headers

Example of Delivery Status Report 1

```
Date: Thu, 7 Jul 1994 17:16:05 -0400
From: Mail Delivery Subsystem <MAILER-
DAEMON@CS.UTK.EDU>
Message-Id: <199407072116.RAA14128@CS.UTK.EDU>
Subject: Returned mail: Cannot send message for 5
days
To: <owner-info-mime@cs.utk.edu>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-
status;
      boundary="RAA14128.773615765/CS.UTK.EDU"

--RAA14128.773615765/CS.UTK.EDU

The original message was received at Sat, 2 Jul 1994
17:10:28 -0400
```



```
from root@localhost

   ----- The following addresses had delivery
problems -----
<louisl@larry.slip.umd.edu>  (unrecoverable error)

   ----- Transcript of session follows -----
<louisl@larry.slip.umd.edu>... Deferred: Connection
timed out
      with larry.slip.umd.edu.
Message could not be delivered for 5 days
Message will be deleted from queue

--RAA14128.773615765/CS.UTK.EDU
content-type: message/delivery-status

Reporting-MTA: dns; cs.utk.edu
```

```
Original-Recipient: rfc822;louisl@larry.slip.umd.edu
Final-Recipient: rfc822;louisl@larry.slip.umd.edu
Action: failed
Status: 4.0.0
Diagnostic-Code: smtp; 426 connection timed out
Last-Attempt-Date: Thu, 7 Jul 1994 17:15:49 -0400

--RAA14128.773615765/CS.UTK.EDU
content-type: message/rfc822

[original message goes here]
--RAA14128.773615765/CS.UTK.EDU--
```


Delivery Status Report fields - 1

per-message-fields =

`[ original-envelope-id-field CRLF ]`	Envelope identifier from request.
`reporting-mta-field CRLF`	MTA which attempted to perform the delivery or relay.
`[ dsn-gateway-field CRLF ]`	Name of gateway which transformed foreign delivery report.
`[ received-from-mta-field CRLF ]`	MTA from which the message was received.
`[ arrival-date-field CRLF ]`	Arrival date to reporting MTA.
`*( extension-field CRLF )`	

Delivery Status Report fields - 2

per-recipient-fields =

`[ original-recipient-field CRLF ]`	Original recipient when sent.
`final-recipient-field CRLF`	Final recipient to whom delivery status is reported.
`action-field CRLF`	failed, delyaed, delivered, relayed (to non-DSA environment), expanded.
`status-field CRLF`	Status code (RFC 1893) (DIGIT "." 1*DIGIT "." 1*3DIGIT
`[ remote-mta-field CRLF ]`	Name of MTA which reported to reporting MTA.
`[ diagnostic-code-field CRLF ]`	Sometimes less preciste diagnostic code from remote MTA.
`[ last-attempt-date-field CRLF ]`	Time of last delivery attempt.
`[ final-log-id-field CRLF ]`	Log entry in final MTA logs.
`[ will-retry-until-field CRLF ]`	Time when delivery attempts will stop.
`*( extension-field CRLF )`	

Example of a Message heading

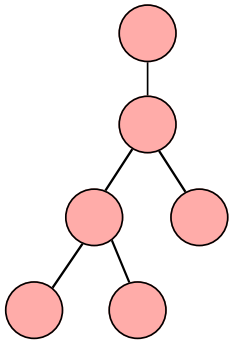
```
From comp.protocols.iso.x400-outbound-request@ics.
uci.edu Wed Nov  4 04:16 MET 1992
Received: from sunic.sunet.se by heron.dafa.se
(16.6/SiteCap-3.0)
          id AA09605; Wed, 4 Nov 92 04:16:24 +0100
Received: from USENET by q2.ics.uci.edu id aa14789; 3 Nov
92
          13:46 PST
From:        "Paul.Rarey" <prarey@ssf-sys.dhl.com>
Subject:     Re: X400 address
Message-ID:  <921103134349.16483@maverick.ssf-
sys.DHL.COM>
Encoding: 35 TEXT, 12 TEXT SIGNATURE
X-Mailer: Poste 2.0
Date:        3 Nov 92 21:46:13 +00:00
To:       mhsnews@ics.uci.edu,
          Piet Beertema <mcvax!cwi.nl!piet@uunet.uu.net>
cc:       ifip65@ics.uci.edu
```

Some Internet message heading fields

Received
From
To
CC
Bcc
Message-ID
Reply-To
In-Reply-To
References
Date
Subject

Usenet News

Newsgroups
Followup-To
Control



Group sending:

```
From:  Nils Nielsson <nilsn@foo.bar>
To:    Flower-lovers:
   Mary Smith <marys@foo.bar>,
   Eliza Brown <elizab@foo.bar>;
Subject: Growing roses indoors
```

Outbox/Inbox/Folder listing:

John Svensson	22 Dec 2005	This is a subject of a message
Mary Smith <m	23 Dec 2005	Growing roses indoors
Flower-lovers:	23 Dec 2005	Growing roses indoors

Sending mail without disclosing recipients to each other:

```
From:  Nils Nielsson <nilsn@foo.bar>
To:    My love:;
Bcc:   Mary Smith <marys@foo.bar>,
       Eliza Brown <elizab@foo.bar>
Subject: You are the only one I love
```

Usage: To hide a very long list

Alternative: Put your own e-mail address into the To: header.

RFC 2822 syntax definitions 1

```
message          =       (fields / obs-fields)
                         [CRLF body]
body             =       *(*998text CRLF) *998text
```

RFC 2822 syntax definitions 2

```
fields          =       *(trace
                         *(resent-date /
                          resent-from /
                          resent-sender /
                          resent-to /
                          resent-cc /
                          resent-bcc /
                          resent-msg-id))
                        *(orig-date /
                        from /
                        sender /
                        reply-to /
                        to /
                        cc /
                        bcc /
                        message-id /
                        in-reply-to /
                        references /
                        subject /
                        comments /
                        keywords /
                        optional-field)
```


RFC 2822 syntax definitions 3

```
from            =       "From:" mailbox-list CRLF

sender          =       "Sender:" mailbox CRLF

reply-to        =       "Reply-To:" address-list CRLF

to              =       "To:" address-list CRLF

cc              =       "Cc:" address-list CRLF

bcc             =       "Bcc:" (address-list / [CFWS]) CRLF
message-id      =       "Message-ID:" msg-id CRLF

in-reply-to     =       "In-Reply-To:" 1*msg-id CRLF

references      =       "References:" 1*msg-id CRLF
```


RFC 2822 syntax definitions 4

```
msg-id          =   [CFWS] "<" id-left "@" id-right ">" [CFWS]

id-left         =   dot-atom-text / no-fold-quote /
                    obs-id-left

id-right        =   dot-atom-text / no-fold-literal /
                    obs-id-right

no-fold-quote   =   DQUOTE *(qtext / quoted-pair) DQUOTE

no-fold-literal =   "[" *(dtext / quoted-pair) "]"
subject         =   "Subject:" unstructured CRLF

comments        =   "Comments:" unstructured CRLF

keywords        =   "Keywords:" phrase *("," phrase) CRLF
```


RFC 2822 syntax definitions 5

```
resent-date     =   "Resent-Date:" date-time CRLF

resent-from     =   "Resent-From:" mailbox-list CRLF

resent-sender   =       "Resent-Sender:" mailbox CRLF

resent-to       =       "Resent-To:" address-list CRLF

resent-cc       =       "Resent-Cc:" address-list CRLF

resent-bcc      =       "Resent-Bcc:" (address-list / [CFWS])
CRLF

resent-msg-id   =       "Resent-Message-ID:" msg-id CRLF
```

RFC 2822 syntax definitions 6

```
trace          = [return]
                 1*received
return         = "Return-Path:" path CRLF
path           = ([CFWS] "<" ([CFWS] / addr-spec) ">" [CFWS]) /
                 obs-path
received       = "Received:" name-val-list ";" date-time CRLF

name-val-list = [CFWS] [name-val-pair *(CFWS name-val-pair)]
name-val-pair = item-name CFWS item-value
item-name      = ALPHA *(["-"] (ALPHA / DIGIT))
item-value     = 1*angle-addr / addr-spec /
                  atom / domain / msg-id
```


RFC 2822 “obs-” features

Proper format:

```
From: "John F. Kennedy" <johnf@white-house.gov>
```

“obs” format:

```
From: John F. Kennedy <johnf@white-house.gov>
```

Syntax definition:

```
mailbox-list  = (mailbox *("," mailbox)) / obs-mbox-list
mailbox       = name-addr / addr-spec
name-addr     = [display-name] angle-addr
display-name  = phrase
phrase        = 1*word / obs-phrase
word          = atom / quoted-string
obs-phrase    = word *(word / "." / CFWS)
obs-mbox-list = 1*([mailbox] [CFWS] "," [CFWS]) [mailbox]
```


Multipurpose Internet Mail Extensions (MIME)

- Multiple objects in one message.
- Unlimited line length and message length.
- Character sets other than IA5 (7-bit ASCII).
- Binary and application-specific files.
- Diagrams, pictures, voice, video, and multimedia.
- References to files, which can be retrieved.
- RFC 1522: Non-ascii characters in message headings

MIME Encoding of Data

Encoding type	Example
7Bit	`Shrimps sandwich` Note that `R}ksm|rg{s` is not 7BIT encoding, it is faulty usage of MIME
Quoted-Printable	R=E4ksm=F6rg=E5s
Base64	`UuRrc232cmflcw0KDQo=`
8Bit	`Räksmörgås`
Binary	`01010010100010100110101 1, etc.`

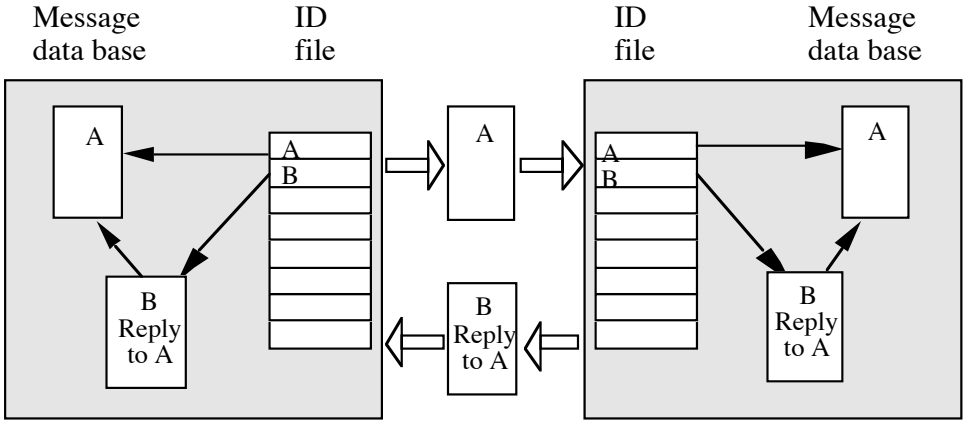
Example of a MIME-Encoded Message-1a

```
Test message containing 8-bit characters.
AE=Ä, OE=Ö.

HEADING
Return-Path: <jpalme@ester.dsv.su.se>
Date: Sun, 26 Sep 1993 18:49:01 +0100 (MET)
From: Jacob Palme DSV <jpalme@dsv.su.se>
Subject: A pine message
To: Lars Enderin <larse@dialog.se>
Message-Id: <3.85.93.A27024-0200000@ester>
Mime-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="1430317162"
```

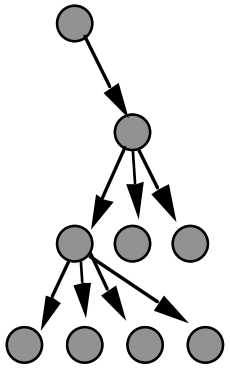


```
PART 1
--1430317162
Content-Type: TEXT/PLAIN; CHARSET=ISO 8859-1
Content-Transfer-Encoding: QUOTED-PRINTABLE

Test message containing 8-bit characters.
AE=3D=80, OE=3D=85.

Test message containing 8-bit characters.
AE=3D=80, OE=3D=85.

PART 2
--1430317162
Content-Type: TEXT/PLAIN; CHARSET=ISO 8859-1
Content-Transfer-Encoding: BASE64

VGVzdCBtZXNzYWdlIGNvbnRhaW5pbmcgOC1iaXQgY2hhcmFjdGVyc
gCwgT0U9hS4K
--1430317162--
```

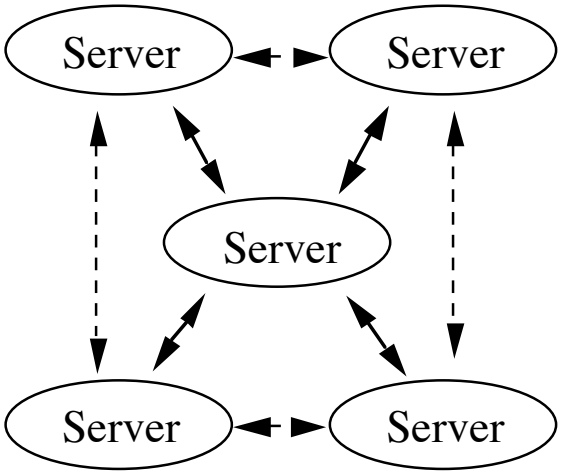

Use of Message-Ids to identify replies

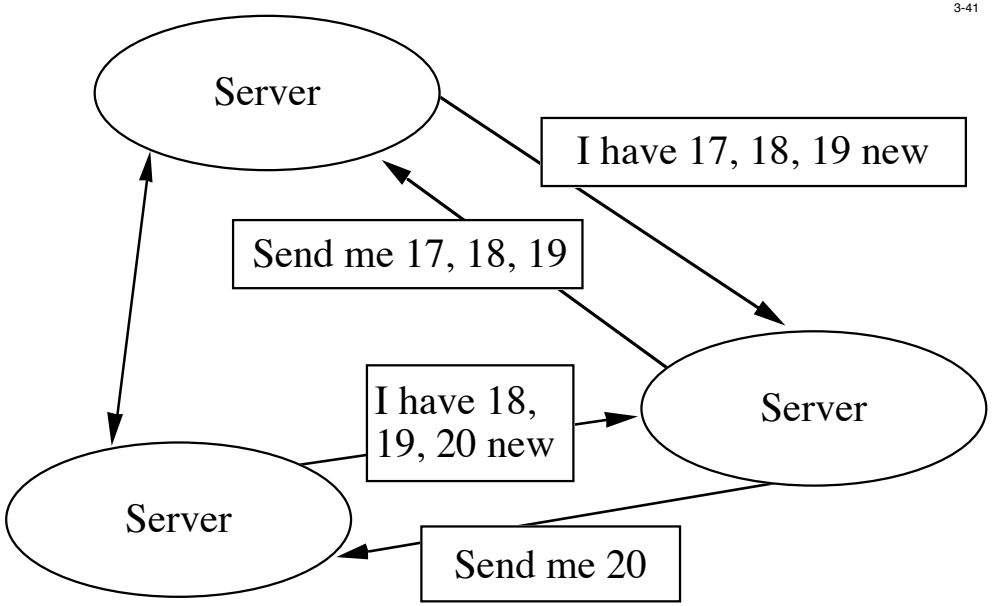


Distribution lists

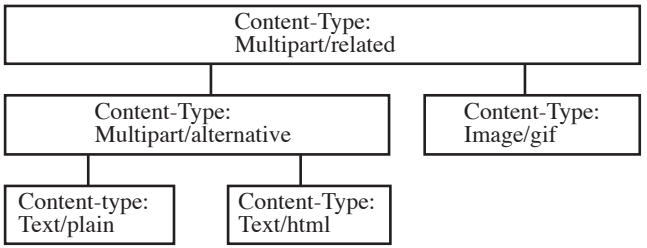


Usenet News distribution method

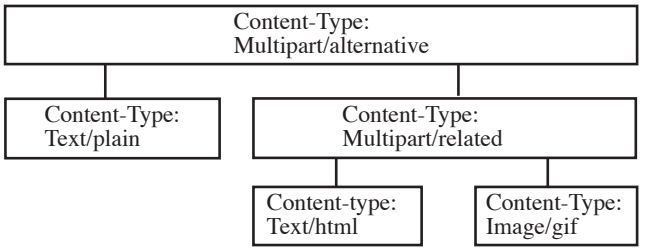




Multipart Messages in MHTML



With the multipart/alternative inside the multipart/related



With the multipart/alternative outside the multipart/related

Multipart/related (HTML with inline objects)

RFC 2110-2112

Method 1

Content-Type: Multipart/related
Content-Type: Text/html Here is a picture <img src="**cid:12345@jp@host.nu**">
Content-Type: Image/gif Content-ID: **:12345@jp@host.nu** VGVzdCBtZXNzYWdlIGNvbnRhaW5pbm

Method 2

Content-Type: Multipart/related
Content-Type: Text/html Here is a picture <img src="**my-picture.gif**">
Content-Type: Image/gif Content-location: **my-picture.gif** VGVzdCBtZXNzYWdlIGNvbnRhaW5pbm

Combining multipart/alternative with multipart/related

Alternative outside

Content-Type: Multipart/alternative
Content-Type: Text/plain Plain text version ...
Content-Type: Multipart/related
Content-Type: Text/html HTML text version ...
Content-Type: Image/gif

Related outside

Content-Type: Multipart/related
Content-Type: multipart/alternative
Content-Type: Text/plain Plain text version...
Content-Type: Text/html HTML text version ...
Content-Type: Image/gif

*:96 Overheads

Part 4: Message delivery protocols (POP and IMAP) Network News (Usenet News)

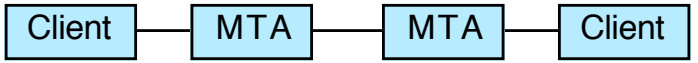
More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

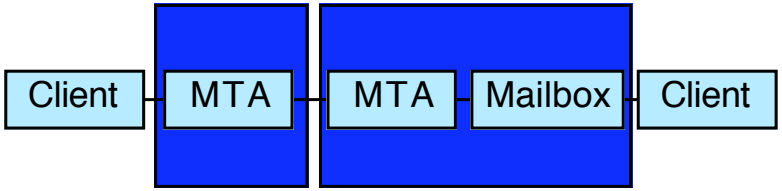
Last update: 2005-12-23 13:03

E-mail model: Where is POP and IMAP?

Traditional mail transport model



Augmented model with mailbox in server



Client-Server Protocols

- Post Office Protocol (POP), a protocol for fast downloading of mail to client software, where the client stores and handles the mail, corresponding to P3 in X.400.
- Interactive Mail Access Protocol (IMAP), a protocol for cases where the user wants to store his messages in the server, and wants to be able to manipulate this storage from client software on his personal computer. IMAP is a more complex protocol than POP.

Post Office Protocol (POP) *(RFC 1939)*

USER	Client identifies mailbox to be downloaded
PASS	Password
STAT	Get number of messages and size of mailbox
LIST N	Return size of message N
LAST	Get highest message number accessed
RETR N	Retrieve a full message
TOP N M	Retrieve only headers and first M lines of message N
DELE N	Delete message
QUIT	Release service
NOOP	See if POP server is functioning
RPOP	Insecure authentication

Example of a POP session

```
S: <waiting for connection on TCP port 109>
C: <open connection to server>
S: +OK POP server ready
C: User larse
S: +OK password required
C: PASS **********
S: +OK mailbox contains 4 messages (4567 octets)
C: STAT
S: +OK 4 4567
C: LIST
S: +OK
S: 1 333
S: 2 906
S: 3 999
S: 4 1111
S: .
```



```
C: TOP 1 1
S: +OK
S: Return-Path: <jpalme@ester.dsv.su.se>
S: Date: Sun, 26 Sep 1995 18:49:01 +0100 (MET)
S: From: Jacob Palme DSV <jpalme@dsv.su.se>
S: Subject: =?iso-8859-1?Q?R=E4ksm=F6rg=E5s?=
S: To: Lars Enderin <larse@dialog.se>
S: Message-Id: <3.85.93.A27024-0200000@ester>
S: Mime-Version: 1.0
S: Content-Type: MULTIPART/MIXED; BOUNDARY="1430317162"
S: --1430317162
S: Content-Type: TEXT/PLAIN; CHARSET=ISO 8859-1
S: Content-Transfer-Encoding: QUOTED-PRINTABLE
S:
S: R=E4ksm=F6rg=E5s
S: .
C: QUIT
S: +OK POP server quitting
C: <close connection>
S: <close connection>
```


Interactive Mail Access Protocol (IMAP), *(RFC 1203)*

- Server can send unsolicited messages to the client.
- More than one transaction can be waiting, identified by transaction ID, asynchronous operation of client and server.
- Each message has a set of properties, which can be retrieved one or more than one at a time. Examples of properties: seen flag, deleted flag.
- To delete a message, set its deleted flag to 1 and perform the expunge command.
- A SEARCH command.
- Bulletin-board facility (shared mailboxes)

IMAP properties:

- More the processing in the server
- Not so much wait for downloading
- More oriented toward keeping the session open
- More flexible than POP

Differences between POP and IMAP

POP	IMAP	Function
Yes	Yes	Download of new mail
Yes	Yes	List of stored messages
Yes	Yes	Peeking at messages
No	Yes	Sort messages into folders on server
No	Yes	Search for messages in the server using different search criteria
No	Yes	Sending operations asynchornously in both directions, with tags to indicate what is answer to what
No	Yes	Run in the background all the time, advice when new messages arrive

Network News Transfer Protocol (NNTP)

RFC 977: Network News Transfer Protocol, February 1986.

RFC 1036: Standard for interchange of USENET messages, January 1987.

Note also UUCP: Alternative to both NNTP and SMTP for communication between servers via asynchronous lines.

NNTP architecture

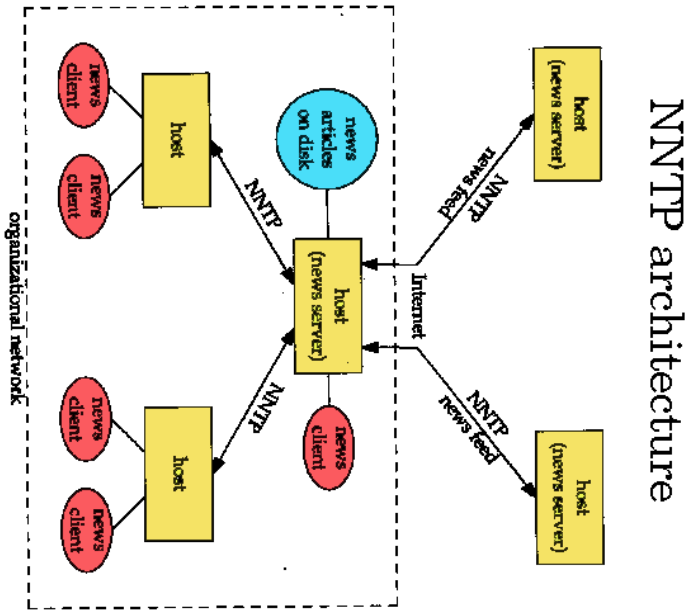


Figure 15.1 Typical news setup.

Picture from Stevens: TCP/IP Illustrated, Volume 3, page 208

Example of an NNTP interaction

```
C:<open connection to news.su.se on port 119>
S:200 news.datakom.su.se InterNetNews NNRP server INN
  1.4unoff3 17-Oct-95 ready (posting ok).
C:group swnet.test                    Go to newsgroup swnet.test
S:211 410 4211 4622 swnet.test        Status info about this
  meeting
C:head 4622                           Get head of message 4622
S:221 4622 <199607282133.OAA17435@infinity.c2.org> head
  Path:news.datakom.su.se!newsfeed.sunet.se!mail2news
  Date: Sun, 28 Jul 1996 14:33:18 -0700 (PDT)
  Message-ID: <199607282133.OAA17435@infinity.c2.org>
  From: jamosen@alpha.c2.org (James Olafsen)
  Subject: **** IPS!*****
  Newsgroups: swnet.test
  Lines: 2
  .
```

Submit an article

```
C: POST
S: 340 Ok
C: From: jpalme@dsv.su.se
   Newsgroups: swnet.test
   Subject: Testing NNTP
   Date: 07 Aug 1996 02:16:40 +0200
   Sender: jpalme@dsv.su.se
   Message-ID: <1-test*jpalme@dsv.su.se>

   This is a test message
   This is the last line of the test message
   .
S: 240 Article posted
C: quit
S: 205
S: <closes the connection>
```


Some NNTP commands

`article [<Message-ID> | <Number>]`	Return text of designated article. If no parameter is given, the next article is returned. The current article pointer is put at the fetched article.
`body [<Message-ID>|<Number>]`	As `article`, but only returns body
`group <newsgroup>`	Go to the designated newsgroup
`head [<Message-ID> | <Number>]`	As `article`, but only returns head
`help`	Lists available commands
`ihave <messageID>`	Informs the server of an available article. The server can then ask for the article or refuse it.
`last`	Sets current article pointer to last message available, returnle number and Message-ID.
`list [active|newsgroups|distributions|schema]`	Returns a list of valid newsgroups in the format: *group last first*
`newgroups <yymmdd`	List newgroups created since a certain

`hhmmss> ["GMT"] [<distributions>]`	datetime. "distributions" can be e.g. *alt* to only get newsgroups in the *alt* category.
`newnews <newsgroups> <yymmdd hhmmss> ["GMT"] [<distributions>]`	List Message-ID of articles posted to one or more newsgroups after a specific time. *newsgroups* can be. e.g. `net.*.unix` to match more than one newsgroups. *distributions* checks for articles which also has this other newsgroup as recipient.
`next`	Current article pointer is advanced. Returns number and Message-ID of current article.
`post`	Submit a new article from a client.
`slave`	Tells the server that this is not a user client, it is a slave server. (May give priority treatment.)
`stat [<Message-ID> | <Number>]`	As `article`, but only returns Message-ID. Used to set the current article pointer.

Network News versus e-mail header formats

The format for Network News headers is based on RFC 822. But there are some differences.

Header	Description
`Newsgroups:`	Comma-separated list of newsgroups to which this article belongs. Example of newsgroup format: alt.sex.fetisches.feet. *Should never occur in e-mail.* Use “Posted-To:” instead!
`Subject:`	Add four characters “`Re.` ” for replies. Do not change subject in replies.
`Message-ID:`	Mandatory in Network News, and must be globally unique.
`Path:`	Path to reach the current system, e.g. abc.foo.net!xyz!localhost. E-mail path format also permitted. Compare to `Received:` and `Return-Path` in e-mail.
`Reply-To:`	In news: Where replies to the author should be sent. In e-mail: Ambiguous.
`Followup-To:`	Where replies to newsgroup(s) should be sent.
`Expires:`	Suggested expiration date.

`References:` Message-ID-s of previous areticles in the same thread. Should always contain first and last article in thread. Compare to e-mail: Usually only immediately preceeding messages..

`Control:` Not used in e-mail. Communication with servers. Body or subject contains command. Subject begins with "`cmsg`".

`cancel`	Delete physically a previously sent article.
`ihave`	Host telling another host of available new articles.
`sendme`	Host asking for articles from another host.
`newgroup`	Name of new group, plus optional word `moderated`.
`rmgroup`	Remove a newsgroup. Requires `approved`.
`sendsys`	Send the sys file, listing neighbours and newsgroups to be sent to each neighbour.
`version`	Version of software wanted in reply.
`checkgroups`	List of newsgroups and descriptions, used to check if list is correct.

`Distribution:` Not used in e-mail. Limits distribution to certain geographical/organizational area. Example: Distribution: se,

no.

`Organization:` Of sender.

`Keywords:` For filtering.

`Summary:` Brief summary.

`Approved:` Required for message to moderated group. Added by the moderator, contains his e-mail address. Also required for certain control messages.

`Lines:` Count of lines of the message body.

`Xref:` Numbers of this message in other newsgroups. Only for local usage in one server. Example: Xref: swnet.risk:456 swnet.sunet:897

MIME versus UUENCODING in e-mail and Network News

MIME is not yet much used in Network News, but is coming slowly.

UUENCODE is an older alternative to Base64 for encoding of binary data, still much used in Usenet News. Base64 is more secure.

Message size restrictions means that large binary files are often split into several articles, intelligent News clients can automatically find and combine them. (The MIME protocol for partitioning of large files is not much used yet in Usenet News.)

*:96 Overheads

Part 5: FTP More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 04-06-15 17.35

File Transfer Protocol (FTP)

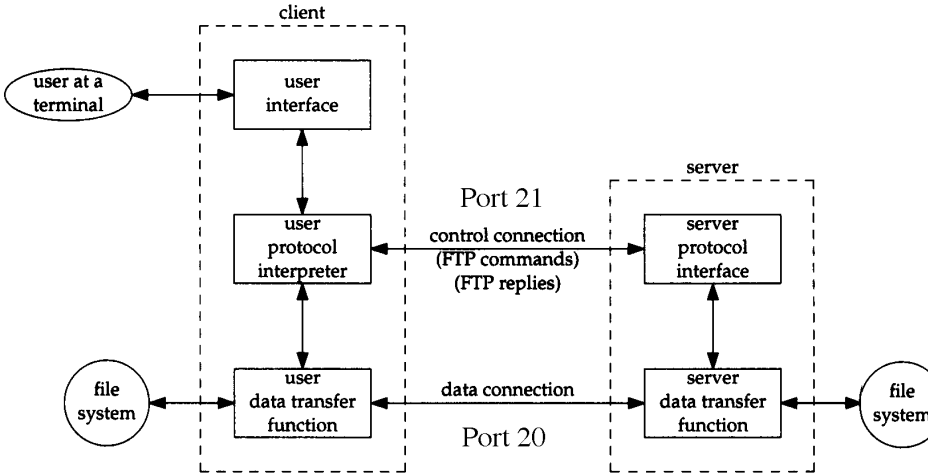
- Login on the client computer.
- Establish a connection between two hosts.
- Log in, using password, on an account at the server.
- Can then perform commands to list files, change directory, send and retrieve files but only files accessible to that account at that server.
- No jumping from computer to computer as in Gopher and World Wide Web.

FTP file types

Only the formats in italics below are in reality usually used today:

File type:	*ASCII*, EBCDIC, *Image (Binary),* Local (transforms byte sizes)
Format:	*Nonprint,* Telnet vertical format control, Fortran carriage control
Structure:	*Continuous stream of bytes,* record structure, page structure (historical)
Transmission mode:	*Stream,* Block, Compressed (very simple run length compression)

FTP architecture channels and ports



Picture from Stevens: TCP/IP Illustrated, volume 1 page 420

Some FTP commands

Command	Description
RETRIEVE (RETR)	Transfer a file from the server to the client
STORE (STOR)	Transfer a file from the client to the server.
STORE UNIQUE (STOU)	Transfer a file from the client to the server, give it a new unique name on the server.
APPEND (APPE)	Transfer a file from the client to the server, if the file already exists, append the file at the end of the existing file.
RESTART (REST)	Start transfer from a checkpoint of the file instead of from the start, applies to the immediately following transfer command.
RENAME FROM (RNFR)	Old path name of a file to be renamed.
RENAME TO (RNTO)	New path name of a file to be renamed
ABORT (ABOR)	Abort the previous FTP service command.
DELETE (DELE)	Delete a file at the server.
REMOVE DIRECTORY (RMD)	Remove a directory at the server.
MAKE DIRECTORY (MKD)	Create a directory at the server.
LIST (LIST)	List the files in a directory in the server.

FTP error codes

First digit:

1yz Preliminary positive
2yz Positive
3yz Positive intermediate
4yz Transient negative
5yz Permant negative

Second digit:

x0z Syntax error
x1z Information
x2z Connection control
x3z Authentication, accounting (login)
x4z Unspecified
x5z Filesystem status

Examples:

125 Data connection open, transfer starting
200 Command OK
214 Help message for human user
331 Username OK, password required
425 Cannot open data connection
500 Syntax error, unrecognized command
501 Syntax error, invalid arguments
502 Unimplemented MODE type

Table from Stevens: TC P/IP Illustrated, volume 1 page 420

Anonymous FTP

User name: Anonymous

Password: <your e-mail address>

Access: Restricted access to public files only

Example of an FTP dialogue with the UNIX line-oriented user interface

```
/home/ester/dsv/dsv-jp> ftp sunic.sunet.se
Connected to sunic.sunet.se.
220 sunic.sunet.se FTP server (Version wu-2.4(3) Sun Oct 1 17:47:58 MET
1995) ready.
Name (sunic.sunet.se:jpalme): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: jpalme@dsv.su.se
230 Guest login ok, access restrictions apply.
ftp> cd rfc
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 76779
-rw-r--r--  1 root     wheel       62646 Jan  7  1996 .cache
-rw-r--r--  1 root     wheel      262206 Jan  4  1996 .cache+
drwxr-xr-x  2 root     wheel         512 Jul 20 04:10 .cap
-rw-r--r--  1 root     daemon      74112 Jul 31 04:10 .mirror
-rw-r--r--  1 root     daemon        116 Nov  6  1995 .rfc-index
drwxr-xr-x  2 1625     daemon        512 Mar  7 13:09 .waisindexes
-r--r--r--  1 root     wheel        6319 Jul 29 19:25 fyi-index.txt
```

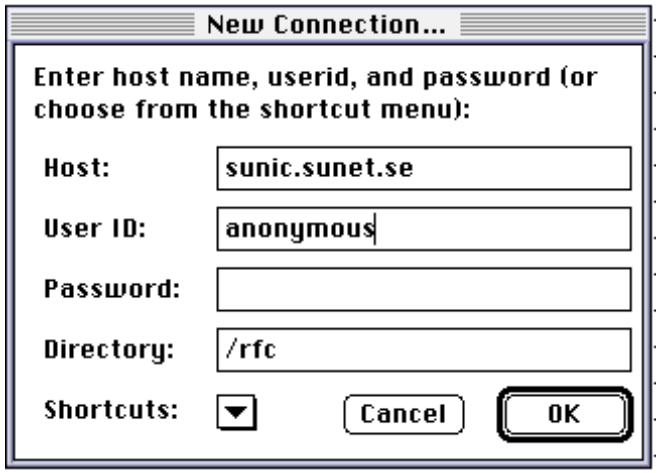


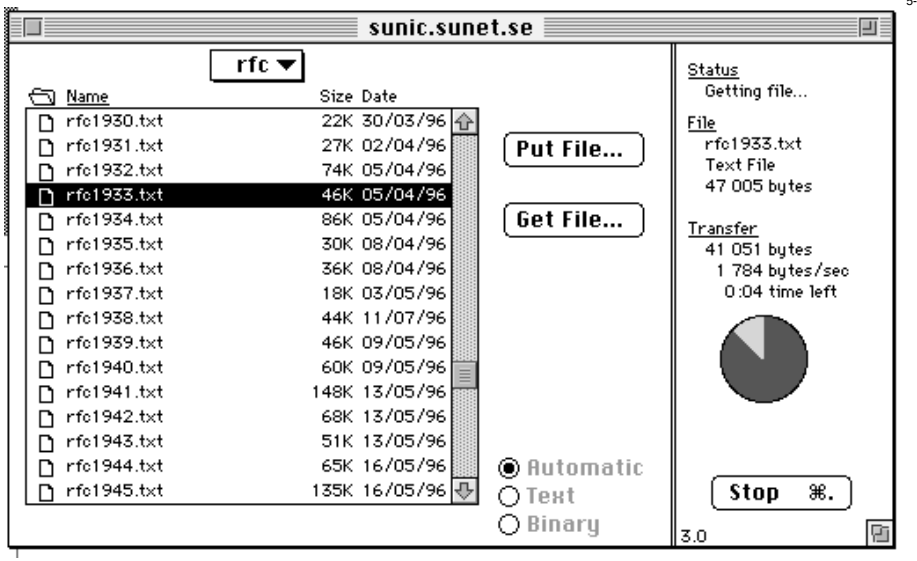
```
drwxr-xr-x  2 root     wheel         512 Nov  8  1995 rfc-editor
-r--r--r--  1 root     wheel      267821 Jul 30 16:14 rfc-index.txt
lrwxrwxrwx  1 root     daemon         11 Apr 27 04:11 rfc-instructions.
-r--r--r--  1 root     wheel       12062 Mar 23 15:30 rfc-retrieval.txt
-r--r--r--  1 root     wheel       12056 Jan 13  1996 rfc-retrieval.txt
-r--r--r--  1 root     wheel        3348 Oct 15  1992 rfc10.txt
-r--r--r--  1 root     wheel      315315 Oct 15  1992 rfc1000.txt
ftp> get rfc1933.txt
200 PORT command successful.
150 Opening ASCII mode data connection for rfc1933.txt (47005 bytes).
226 Transfer complete.
local: rfc1933.txt remote: rfc1933.txt
48240 bytes received in 0.2 seconds (2.4e+02 Kbytes/s)
ftp> help
Commands may be abbreviated.  Commands are:

!               cr              macdef          proxy           send
$               delete          mdelete         sendport        status
account         debug           mdir            put             struct
append          dir             mget            pwd             sunique
ascii           disconnect      mkdir           quit            tenex
bell            form            mls             quote           trace
binary          get             mode            recv            type
bye             glob            mput            remotehelp      user
case            hash            nmap            rename          verbose
```



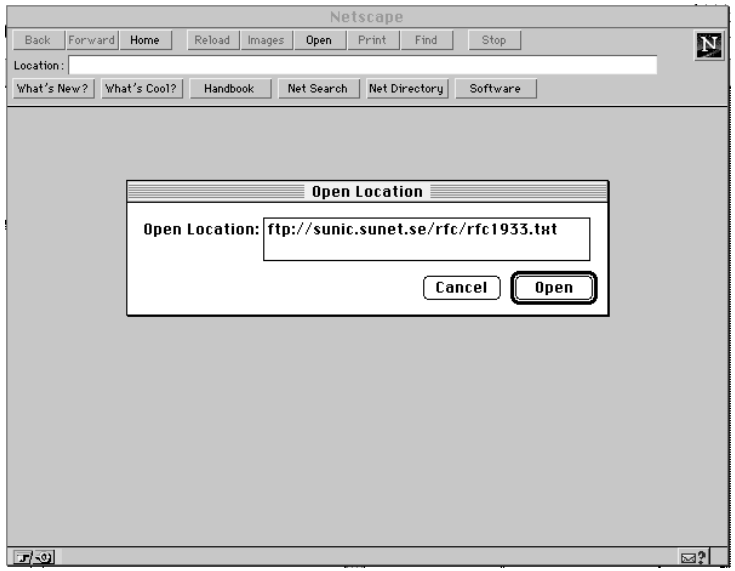
```
cd              help            ntrans          reset           ?
cdup            lcd             open            rmdir
close           ls              prompt          runique
```

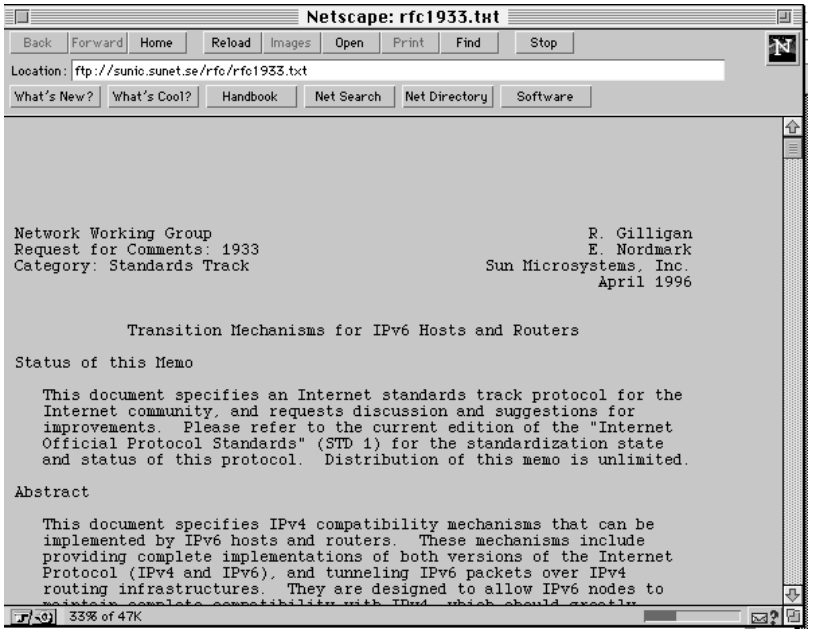
Example of an FTP dialogue using a custom graphical program





Example of FTP access using a web browser





*:96 Overheads

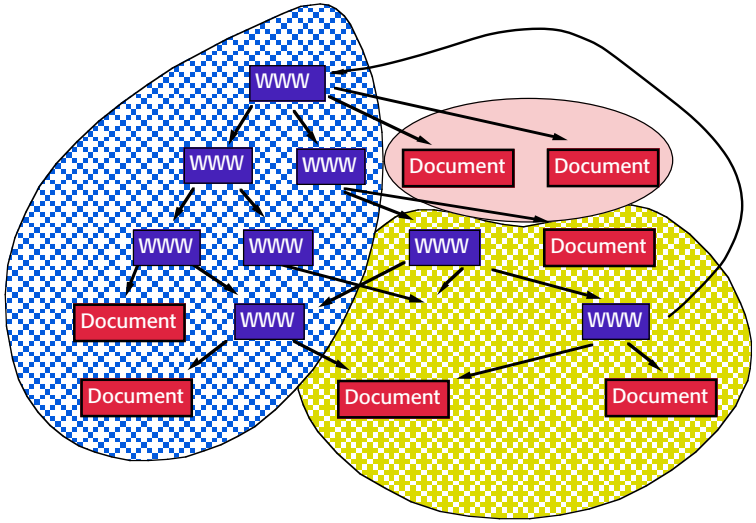
Part 6a: World Wide Web, Hypertext Markup Language (HTML)

More about this course about Internet application protocols can be found at URL:

http://www.dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html

Last update: 14 Jan 2005

Distributed data base in WWW



Some common web page formatting standards

HTML	Hypertext Markup Language	Mostly used. Does not give an exact precise specification of how a page will look on the screen. Actual page view depens on the size of the browser window, browser settings, etc.
XHTML	XML HTML	XML compliant variant of HTML.
Javascript, ECMAscript, DOM	DOM = Document Object Model	Programmable interface to web page, usually embedded in HTML code.
GIF	Graphics Interchange Format	Max 256 colors, but colors can be chosen to a set suitable for each picture. Patent and copyright problems.
JPEG	Joint Photographic Experts Group	Kraftig komprimering av fotografier och målningar. Förstörande komprimering.
Java, Flash		Advanced formats, do not work for all users.
PDF	Portable Document Format	Exact rendition of documents for printing.

Do you need to learn HTML?

No, you can use

- HTML editors, which produce HTML automatically. Some HTML editors hide almost all HTML from the user. Example: Adobe Pagemill. Other editors are ordinary text editors with enhanced functions to aid writing of HTML. Example: Pagespinner (developed by a student at DSV) and BBEDIT.

- Converters between word processing formats and HTML. Examples: RtfToHTML, converters from DataViz, converters provided by the developers of word processors like Microsoft Word and Word Perfect as included or additional functionality.

HTML tools (editors, converters, etc.) often do not support full HTML. And if you use HTML editors to edit old HTML documents, they may even remove or munge HTML codes they do not understand! My experience is that existing tools are useful, but you still have to go into the HTML source now and them.

You may need to learn HTML for:

- To understand how it works behind the scenes.
- To use HTML functions or perform formatting not supported by the HTML editor or converter you are using.
- To able to write programs which generate HTML, for example for servers which produce HTML on-the-fly based on user requests.

Software tools for Web page creation

Software tools for HTML creation can roughly be categorized as follows:

Category	Examples
Text editor with special HTML support	BBEDIT, Page Spinner
Microsoft particular editor (not good for producing documents to be viewed with other browsers than Microsoft Explorer)	Frontpage
Advanced WYSWYG HTML editor may include features like page layout or automatic java creation	NetObjects Fusion, Dreamweaver, Golive
Site support tools, with aid to get an overview of your site, check for faulty links, etc.	Sitemill
Conversion tools from other formats	RTFtoHTML, built into many other programs
Tools to create web graphics	Photoshop, Fireworks, ImageReady

HTML versus Postscript and PDF

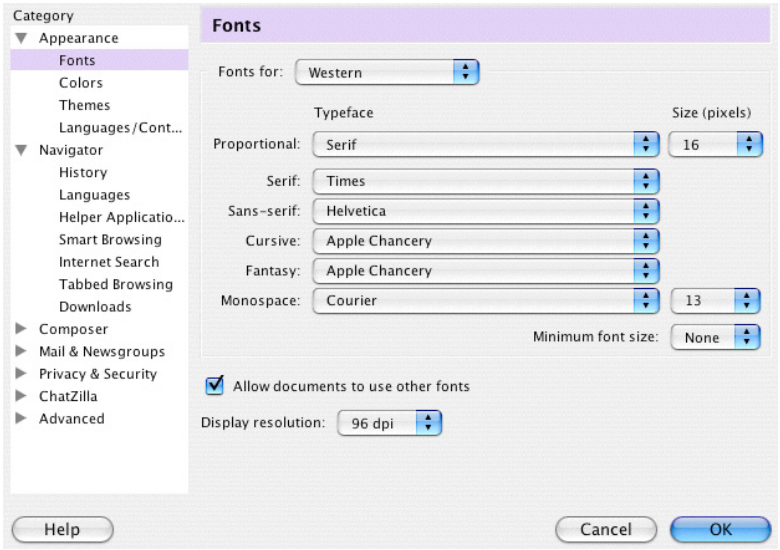
Postscript and PDF (Adobe Portable Document Format, used by Adobe Acrobat) are formats that exactly specify how a page will look like, fonts, sizes, distances, etc.

This is not done by HTML. HTML specifies that a certain part of a document is for example:

- A header
- subheader at level 2, 3, etc.
- Emphasized text (often italic text)
- Strongly empathized text (often bold text)
- A hyperlink
- A list of numbered or bulleted items

Which font is to be used for the different heading levels, for ordinary text, etc. is specified by so-called “Style sheets”, which are often set by the developer of the web browser or by its users. Example from Netscape:

Example of user configuration of style sheets with Netscape



Differences between web browsers

The same HTML document may look different when displayed with different web browsers for many reasons:

- Different style sheets (font and size for different kinds of elements)
- Different interpretation of unclear parts of the HTML standard
- Different sets of HTML elements supported. Many web browsers, especially Mozilla and Microsoft Explorer, have their own non-standard additions to HTML. Also, all web browsers do not support all standardized functions.

Examples of features not supported by all web browsers:

- Blinking text: Non-standard and much disliked Netscape addition to HTML.
- Tables and frames.
- Graphics. There are text-only browsers (Lynx), special browsers for disabled people and users can often configure their browsers to not download graphics automatically.

Most web browsers will handle HTML elements it does not recognize by simply ignoring it. Most web browsers are highly capable of accepting erroneous and non-standard HTML text, but what they do with such text varies from browsers to browser.

Format of HTML text

HTML documents are plain text documents containing a combination of text and special HTML commands, called HTML elements.

Example of a command which has a start and an end tag:

```
<STRONG>Text to display strongly</STRONG>
```

Example of a command which has only a start tag:

```
<BR>This is the beginning of a new paragraph.
```

HTML commands are case-insensitive:

"<STRONG>", "" and "<StroNg>" are identical commands.

Structure of HTML documents

This line should always start a document and indicate which HTML style sheet the document uses.

Indicate which version of HTML you are using, for example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//WebTechs//DTD Mozilla HTML//EN">
```

Start of HTML.

```
<HTML>
```

Head contains information general for the whole document.

```
<HEAD>
<TITLE>This is the title of this document</TITLE>
</HEAD>
```

Body contains the text.

```
<BODY>
<H1>This is the main heading</H1>
This is the text of the first paragraph.
</BODY>
```

End of HTML.

```
</HTML>
```


Why some characters need special encoding

HTML markup:

```
<TT>If x &gt; 3 then exit.</TT>
<P>
<TT>The word
&lt;strong&gt;<strong>strong</strong>&lt;/strong&gt; is
rendered in a strong way.</TT>
```

Shows on the screen as:

```
If x > 3 then exit

The word <strong>strong</strong> is rendered in a
strong way.
```

Characters which need special encoding

Glyph	Symbolic coding	Numeric coding
<	<	<
>	>	>
&	&	&
"	"	"

Examples of encoding of 8-bit characters (not absolutely necessary for ISO-8859-1) Note: Case-sensitive!

Glyph	Symbolic coding	Numeric coding
<non-breaking-space>		
å	å	å
ä	ä	ä
ö	ö	ö
Å	Å	Å
Ä	Ä	Ä
Ö	Ö	Ö
©	©	©
®	®	®

Important elements in the head of a HTML document

Start of head	`<HEAD>`
Title, shown by many browsers at the top of the window. Important for search engines.	`<TITLE>This is the title of this document</TITLE>`
Agreed but not very much used way of indicating author of the document.	`<LINK REV=MADE HREF= "mailto:jpalme@dsv.su.se">`
Description to be shown by web indexers instead of the first lines of the document	`<META name="description" content="This is the home page for Jacob Palme.">`
Words not in the document but which should be indexed by web indexers and important index words for this document.	`<META name="keywords" content="Sweden Stockholm Computer Science">`
End of head.	`</HEAD>`

next page: Robots exclusion

WWW Robots Exclusion

The Robots Exclusion Protocol

A Web site administrator can indicate which parts of the site should not be visited by a robot, by providing a specially formatted file on their site, in http://.../robots.txt.

The Robots META tag

A Web author can indicate if a page may or may not be indexed, or analysed for links, through the use of a special HTML META tag. The remainder of this pages provides full details on these facilities.

Example of the Robots Meta tag:

```
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
```

For more information, see URL:

http://info.webcrawler.com/mak/projects/robots/exclusion.html

Next page: New line markup

New line in HTML markup and in displayed text

HTML markup	Displayed text

`Jack and Jill¶`
`went up a hill.¶`
`Jack fell down,¶`
`and Jill came tumbling after.`
	Jack and Jill went up a hill. Jack fell down, and Jill came tumbling after.

`Jack and Jill went up a hill. <P> Jack fell down, and Jill came tumbling after.`	Jack and Jill went up a hill. Jack fell down, and Jill came tumbling after.
`Jack and Jill went up a hill.   Jack fell down, and Jill came tumbling after.`	Jack and Jill went up a hill. Jack fell down, and Jill came tumbling after.

`<PRE>Jack and Jill¶`
`went up a hill.¶`
`Jack fell down,¶`
`and Jill came tumbling after.`
`</PRE>`
	`Jack and Jill`
	`went up a hill.`
	`Jack fell down,`
	`and Jill came tumbling after.`

Note: New line is significant, in `<PRE>`, and elsewhere too, it becomes a space!!

Next page: HTML headings

HTML headings

`<H1>Top level heading</H1>`	**Top level heading**
`<H2>Level 2 heading</H2>`	**Level 2 heading**
etc. until	
`<H6>Level 6 heading</H6>`	**Level 6 heading**

Different browsers might display the same heading in different formats, for example use italics, bold face, indentation and space before and after the heading in different ways than other browsers.

Good practice rules says that headings must be put in natural order, for example H1, H2, H3, H3, H2; H3 but not H2, H1, H3, H1. This is of special value for sight-impaired people reading the web. If you do not like the size of these headers, use CSS to specify the size you want, instead of using the wrong header.

Paragraph separation

HTML element	Description
<P> (.. </P>)	Paragraph break, also used to associate attributes to a paragraph.
<DIV> .. </DIV>	Like a paragraph, but less distance between lines. Used to associate attributes with a section of code.
 	Line break, less distance between lines.
<HR>	Horizontal rule (line).

Paragraph alignments

`<P ALIGN=LEFT>Left alignment`	Left alignment
`<P ALIGN=CENTER>Centered text`	Centered text
`<P ALIGN=RIGHT>Right alignment`	Right alignment

Comparison of different ways of centering text

before `<P ALIGN=CENTER>`This text is centered.</P> after	before This text is centered. after
before `<BR ALIGN=CENTER>`This text is not centered. after	before This text is not centered. after
~~before~~ ~~` <CENTER>`This text is centered.`</CENTER> `~~ ~~after~~	before This text is centered. after
~~before~~ ~~`<CENTER>`This text is centered`<CENTER>`~~ ~~after~~	Different rendering with Netscape and Explorer
before `<DIV ALIGN=CENTER>`This text is centered.`</DIV>` after	before This text is centered. after

Emphasis within a sentence

`It is very <EM>important</EM> to look down first.`	It is very *important* to look down first.
`It is very <I>important</I> to look down first.`	It is very *important* to look down first.
`It is very <STRONG>important</STRONG> to look down first.`	It is very **important** to look down first.
`It is very <B>important</B> to look down first.`	It is very **important** to look down first.

EM and STRONG is regarded as more good practice than I and B.

Do not combine two kinds of emphasis.

```
It is very <EM><B>important</EM></B> to look down first
```

may not turn out the intended way.

Other special word renderings

`The best book I have ever read is <CITE>Passage of arms</CITE> by Eric Ambler.`	The best book I have ever read is *Passage of arms* by Eric Ambler.
`The command<KBD> RM * </KBD>will delete all your files.`	The command `RM *` will delete all your files.
`The command<TT> RM * </TT>will delete all your files.`	The command `RM *` will delete all your files.
`Einstein found that<CODE> e=mc**2 </CODE>in his theory of special relativity.`	Einstein found that `e=mc**2` in his theory of special relativity.
`The only English word containing the sequence <SAMP>mt</SAMP> is <CITE>dreamt</CITE>.`	The only English word containing the sequence `mt` is *dreamt*.
`Type <SAMP>html-check <VAR>file</VAR> | more</SAMP to check <VAR>file</VAR> for markup errors.`	Type `html-check` *file* `| more` to check *file* for markup errors.

Address

`<ADDRESS>` `Newsletter editor ` `J.R. Brown ` `JimquickPost News,` `Jimquick, CT 01234 ` `Tel (123) 456 7890` `</ADDRESS>`	*Newsletter editor* *J.R. Brown* *JimquickPost News, Jimquick, CT 01234* *Tel (123) 456 7890*

Blockquote

`I believe that it was Churchill who said:`
`<BLOCKQUOTE>Never have so few men done so much for so many`
`</BLOCKQUOTE>commeting on the British Air Force during the second world war.`
	I believe that it was Churchill who said:
	> Never have so few men done so much for so many
	commenting on the British Air Force during the second world war.

Unordered list

`<UL>`
`<LI>First item`
`<LI>Second item<P>`
`second paragraph of second item<LI>Third item</UL>`
	• First item
	• Second item
	second paragraph of second item
	• Third item

Ordered list (note that nesting is allowed)

```
<OL>
<LI>Click the Web button to open URI window.
<LI>Enter the URI number in the text field of the Open URI
window. The Web document you specified is displayed.
  <OL>
   <LI>substep 1
   <LI>substep 2
  </OL>
<LI>Click highlighted text to move from one link to another.
</OL>
```

might be rendered as

(1) Click the Web button to open URI window.

(2) Enter the URI number in the text field of the Open URI window. The Web document you specified is displayed.

   (a) substep 1

   (b) substep 2

(3) Click highlighted text to move from one link to another.

Definition list

```
Now follows a definition list, a list of terms with their
definition:
<DL>
  <DT>STAT<DD>Get number of messages and size of mailbox
  <DT>LIST N<DD>Return size of message N
  <DT>RETR N<DD>Retrieve a full message
  <DT>TOP N M<DD>Retrieve only headers and the first N lines
</DL>
This is after the end of the list
```

Now follows a definition list, a list of terms with their definition:

LIST N
: Return size of message N

RETR N
: Retrieve a full message

TOP N M
: Retrieve only headers and the first N lines

This is after the end of the list

Anchor (Hyperlink)

(1) relative links within the same document

```
<H2>Table of contents</H2>

<A HREF="#anchor1">Heading of chapter 1</A><BR>
<A HREF="#anchor2"> Heading of chapter 2</A><BR>
<A HREF="#anchor3"> Heading of chapter 3</A><BR>

<H2><A NAME="anchor1"> Heading of chapter 1 </A></H2>
Text of chapter 1

<H2><A NAME="anchor2"> Heading of chapter 2 </A></H2>
Text of chapter 2

<H2><A NAME="anchor3"> Heading of chapter 3 </A></H2>
Text of chapter 3
```


Table of contents

Heading of chapter 1

Text of chapter 1

Heading of chapter 2

Text of chapter 2

Heading of chapter 3

Text of chapter 3

Anchors – (2) links between documents

```
Further discussion on this memo should be done through the mailing
list
<A HREF="mailto:MHTML@SEGATE.SUNET.SE">MHTML@SEGATE.SUNET.SE</A>.
To subscribe to this list, send a message to
<A HREF="mailto:LISTSERV@SEGATE.SUNET.SE">LISTSERV@SEGATE.SUNET.SE</A>
which contains the text<BR>
<TT>SUB MHTML &lt;your name (not your e-mail address)&gt;</TT><P>

Archives of this list are available by anonymous ftp from
<A HREF="ftp://SEGATE.SUNET.SE/">FTP://SEGATE.SUNET.SE</A>
in the directory
<A HREF="ftp://SEGATE.SUNET.SE/lists/mhtml/">/lists/mhtml</A>.
The archives are also available by e-mail. Send a message to
<A HREF="mailto:LISTSERV@SEGATE.SUNET.SE">LISTSERV@SEGATE.SUNET.SE</A>
with the text <TT>INDEX MHTML</TT> to get a list of the archive
files, and then a new message <TT>GET &lt;file name&gt;</TT> to
retrieve the archive files.<P>

You can also browse the archives by http from
<A HREF="HTTP://segate.sunet.se/archives/mhtml.html">
HTTP://segate.sunet.se/archives/mhtml.html</A>.
```

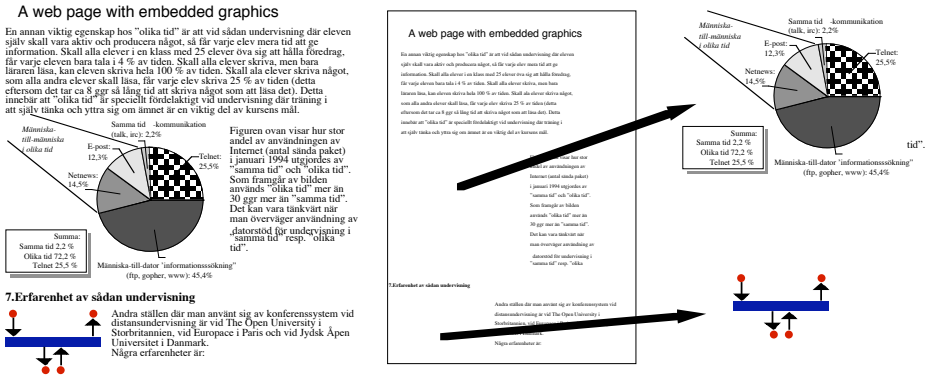

Rendering of the text on the previous slide

Further discussion on this memo should be done through the mailing list MHTML@SEGATE.SUNET.SE. To subscribe to this list, send a message to LISTSERV@SEGATE.SUNET.SE which contains the text
`SUB MHTML <your name (not your e-mail address)>`

Archives of this list are available by anonymous ftp from FTP://SEGATE.SUNET.SE in the directory /lists/mhtml. The archives are also available by e-mail. Send a message to LISTSERV@SEGATE.SUNET.SE with the text `INDEX MHTML` to get a list of the archive files, and then a new message `GET <file name>` to retrieve the archive files.

You can also browse the archives by http from HTTP://segate.sunet.se/archives/mhtml.html.

HTML links to graphics, applets, frames, etc.



Warning: This printout does not say the whole story

These greyscale miniature printouts of the overheads about colour graphics in world wide web often do not tell you the intended story. Many of the overheads cannot be understood unless you view then in colour. But the high cost of printing in colour has forced us to provide these miniature printouts in greyscale only. Providing them in colour would have increased the price of the course documents by about 100 SEK, and the students have told us they are not willing to pay this price to get the printouts in colour.

The overheads are however available, in colour at

http://dsv.su.se/jpalme/internet-course/image-graphics.pdf

A good introductory course on web graphics can be found at

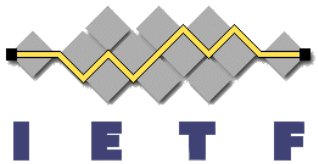
http://builder.cnet.com/webbuilding/0-3883-8-4892140-1.html

Graphics

Graphics in HTML are stored in separate files, one file for each graphic. The main HTML document only contains links to the graphics, not the actual images.

Example:

```
<IMG SRC=
"http://www.ietf.cnri.reston.va.us/images/ietflogo.gif"
 ALT="The IETF logo">
<H1>Internet Engineering Task Force</H1>
```



Internet Engineering Task Force

Object versus bitmapped graphics

Object graphics: Describe a picture with commands like “draw a line of width 1 from point 12,44 to point 12,99” or “draw a circle segment ...” or “fill an area with colour ...”.

Bitmapped graphics: Split image into raster points, indicate colour of each raster point. Usually combined with compression to reduce file size.

Formats for object graphics usually also allow bitmapped graphics within the object graphics.

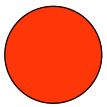
Formats for bitmapped graphics: GIF, JPEG.

Formats for object graphics: Adobe Acrobat, Postscript, PICT.

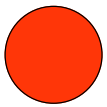
Best supported formats on theWWW is GIF and JPEG, both of which are bitmapped, usually with 72 DPI. This is OK for screeens, but not so good for printing.

Object graphics will automatically get sharper when imaged on screens or printers with high resolution. Bitmapped graphics can never get more sharp than the raster used. Example:

High resolution



Bitmapped 72 DPI

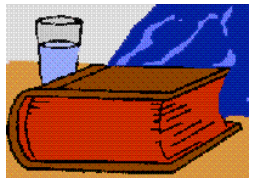


Bitmapped 16 DPI

Dithering (Swedish: Gittring)

Original picture, 72 DPI, 16 bits colour depths

The same picture as shown on a 256-colour (8 bit colour depths) screen



Part of the picture above 3 x enlarged

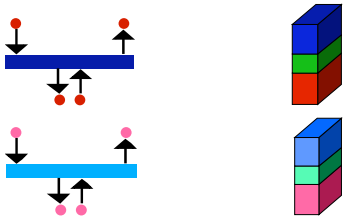
Part of the picture above 3 x enlarged

Some image formats used in the WWW:

Name		Description
Graphic Interchange Format	GIF	Bitmapped image with no-loss compression. Requires UNISYS patent to implement.
Joint Photo-graphers Experts Group	JPEG, JPG	Bitmapped image with information-losing compression based on deficiencies in the human eye. Can for photographs, especially color photograph, give much more compression than GIF without difference visible to the human eye.
Portable Document Format	PDF	Used by Adobe Acrobat. Can render full manuscripts including text and pictures. Proprietary format, reader freeware.
Portable Network Graphics	PNG	Proposal for a new format to replace GIF. See URL `http://raptor.csc.flint.umich.edu/~yost/png-docs/intropng.html` for more info.
Postscript	PS	Can render full manuscript including text and pictures. Large file sizes. Proprietary Adobe format, but widely used by non-Adobe products.

For more info see URL `http://www.berkana.com/class2/media.html`

Two kinds of graphics

Properties	Graphics sharp borders, low number of different colours and large fields with the same colour.	Graphics with many different colours, soft transitions between colours.
Examples	Drawings, diagrams	Photographs, paintings
Quality requirements	Sharp borders, even fields, exact colour matching not important	Many colours, exact colour matching often important
Best web encoding	Usually GIF	Usually JPEG
Examples		

The path from original to web-picture

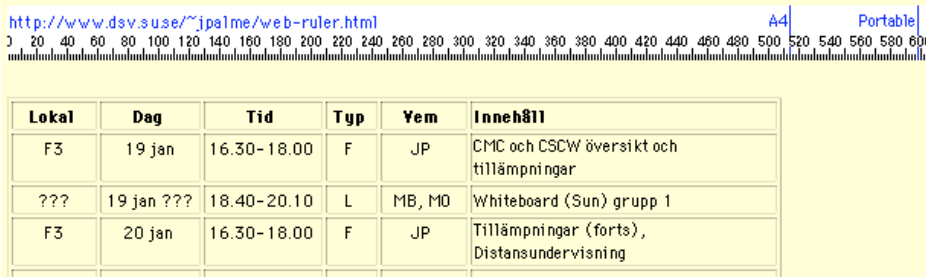
Original	Line drawing	Photo or painting
Common formats:	Illustrator, Freehand or Photoshop	Photoshop, JPEG
Web format:	GIF	JPEG
Conversion to bitmap:	Photoshop, Superpaint, Graphic Converter.	Picture is already in bitmapped format.
Effect of conversion to Web format:	Colour depth reduced to 8 bits or less. Can cause dithering.	Loss of detail, dependent on compression factor.
Also during this conversion:	Bit density usually changed to 72 BPI. Note: Logically always 72 BPI, even if actual screen has higher density.	
Effect when rendering on the screen for a user who has only 256 colours:	Colour may be converted from one to another 256 colour palette. Can cause dithering and sometimes bad distortion.	Colour converted to the palette available on the user screen.

Web pages printable and visible on small screens

Web pages less wide than 514 pixels can be printed on both A4 and US Letter sized paper without loss of information.

Web pages less wide than 600 pixels can be shown on portable computers with 640x480 screen sizes without any need for horizontal scrolling.

At http://www.dsv.su.se/~jpalme/web-ruler.html you can find a ruler, which you can use to test the width of your web pages, as shown by the example below.



For more information see http://www.dsv.su.se/~jpalme/web-ruler.html.

Text flowing around pictures

The setting

The Westin Bonadventure hotel was really impressive. Five 30-floor towers, four in the corners, one in the middle, connected by narrow corridors with 12 scenic elevators.

Calendaring and scheduling

The object of this working group is to produce a standard for sending calendar data (meeting times, requests for bookings, time schedules) across the network.

```
<H2>The setting</H2>
<P><IMG
SRC="http://www.westin.com/graphics/hotels/LABONleft.jpg"
ALIGN="LEFT"></P>
<P>The Westin Bonadventure hotel
was really impressive. Five 30-floor towers, four in the
corners, one in the middle, connected by narrow corridors with
12 scenic elevators.</P>
<P><BR clear="all">
<H2>Calendaring and scheduling</H2>
```

Image maps

It is possible to specify that clicking on different parts of an image has different effects. Such images are called *Image maps*. HTML 2.0 had only so-called *Server-side image maps*, but in HTML 3.2 also *Client-side image maps* were added.

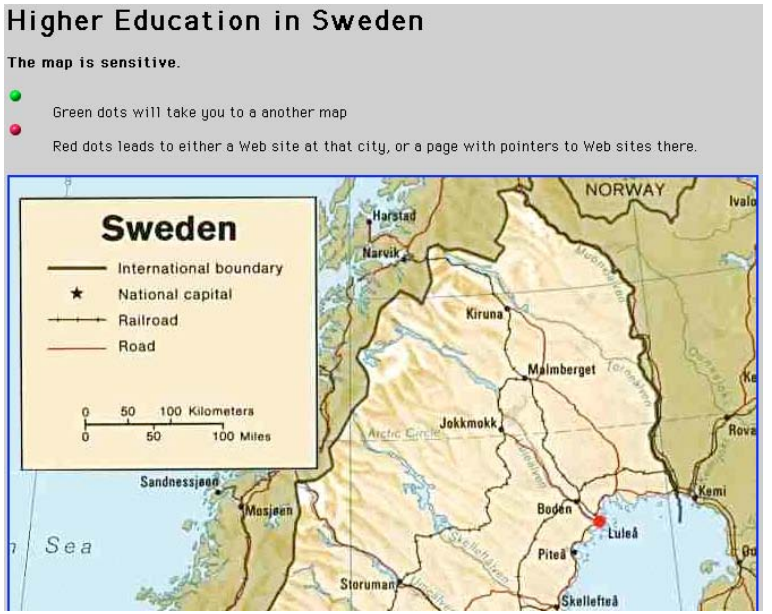
The difference between server- and client-side image maps is where the decision is made what to do.

With server-side image maps, the x- and y-coordinates of the place in the image on which the user clicked are transferred to the server, and the software in the server then decides, based on these coordinates, what to return.

With client-side image maps, the HTML markup contains specifications of which areas of an image will represent different URIs, so that the user's web browser can deduce which URI to use depending on where in the map the user clicked.

This means that with client-side image maps, no special functionality is needed in the server to handle such maps. The individual page designer can thus use image maps without help from the service provider.

More information about image maps see URL
`http://www.berkana.com/class2/maps.html`



Example of a client-side image map.

Example of a server-side image map

The image map below could also have been realized by several separate graphics, one for each command, and with its own URL, or, of course, with plain textual URL-s. What are the pros and cons of these three methods of rendering the same information?

home purchase information exam copies help download library

HTML forms makes HTML into a general user interface generation language

Your name

Password

Postal address

Colour: Blue Green Red Extras: Sound Light Vibration Size: Small

Courier delivery Air mail Surface mail

Registered letter Superior quality

Order Rush order Clear form

HTML forms example; markup:

At URL: HTTP://www.dsv.su.se/~jpalme/test/form-example-1.html

```
<FORM ACTION=ordering-script.cgi
ENCTYPE= "application/x-www-form-urlencoded" METHOD=POST>

Your name<INPUT NAME="name" TYPE=TEXT SIZE="43"
MAXLENGTH="60"><P>

Password<INPUT NAME="PW" TYPE=PASSWORD SIZE="30"><P>

Postal address<TEXTAREA NAME="Address" ROWS="7"
COLS="50"></TEXTAREA><P>

Colour: <SELECT SIZE="3" NAME="Colour">
<OPTION SELECTED>Blue    <OPTION>Green
<OPTION>Red              <OPTION>Yellow
<OPTION>Brown            </SELECT>

Extras: <SELECT NAME="Extras" SIZE="3" MULTIPLE>
<OPTION>Sound <OPTION>Light <OPTION>Vibration </SELECT>

Size: <SELECT NAME="Size">
```



```
<OPTION SELECTED>Small <OPTION>Medium <OPTION>Large
</SELECT><P>

<INPUT TYPE=RADIO VALUE="courier" NAME="delivery">
Courier delivery
<INPUT TYPE= RADIO VALUE="air-mail" NAME="delivery" CHECKED>
 Air mail
<INPUT TYPE= RADIO VALUE="surface" NAME="delivery">
Surface mail<P>

<INPUT TYPE=CHECKBOX NAME="Registered" VALUE="Yes"> Registered
letter

<INPUT TYPE=CHECKBOX NAME="Quality" VALUE="Superior" CHECKED>
Superior quality<P>

<INPUT NAME="submit" TYPE=SUBMIT VALUE="Order">
<INPUT NAME="submit" TYPE=SUBMIT VALUE="Rush order">
<INPUT NAME="reset" TYPE=RESET VALUE="Clear form"><P>

</FORM></BODY></HTML>
```


HTML form attributes

Start of HTML form

Element	Attribute	Description
FORM		Start of a FORM..
	ACTION	The action URI for the form. Default: Base URI of the document>.
	METHOD	GET with no side-effects.
		POST has side-effects.
	ENCTYPE	Media type for encoding sent data.

One-line text input

INPUT		A field for user input.
	TYPE=TEXT	A single line text-entry field.
	MAXLENGTH	Maximum number of characters.
	SIZE	Display space.
	VALUE	Initial value.
	TYPE= PASSWORD	Same as TEXT, but not shown on the screen.

Example:

```
Your name<INPUT NAME="name" TYPE="text" SIZE="43"
MAXLENGTH="60">
```

Rendering:

Your name

Checkbox

INPUT		A field for user input.
	TYPE= CHECKBOX	A checkbox.
	NAME	Symbolic name for group of fields
	VALUE	Portion of the value contributed by this element
	CHECKED	Initial state

Example:

```
<INPUT TYPE="checkbox" NAME="Registered" VALUE="Yes">
Registered letter
<INPUT TYPE="checkbox" NAME="Quality" VALUE="Superior"
CHECKED> Superior quality<P>
```

Rendering:

Radio button

INPUT		A field for user input.
	TYPE= RADIO	A checkbox.
	NAME	Symbolic name for group of fields.
	VALUE	Portion of the value contributed by this element.
	CHECKED	Initial state (can only be set for one in a group).

Example:

```
<INPUT TYPE=RADIO VALUE="courier" NAME="delivery">
Courier delivery
<INPUT TYPE=RADIO VALUE="air-mail" NAME="delivery" CHECKED>
 Air mail
<INPUT TYPE=RADIO VALUE="surface" NAME="delivery">
Surface mail
```

Rendering:

HIDDEN field

INPUT		A field for user input.
	TYPE= HIDDEN	Field which is not displayed to the user, but which returns value when the form is submitted.
	NAME	Symbolic name for group of fields.
	VALUE	Value submitted.

Example:

```
<INPUT TYPE=HIDDEN VALUE="xy654zd" NAME="password">
```

Rendering:

Submit

INPUT		A field for user input.
	TYPE= SUBMIT	A submit button.
	NAME	Symbolic name.
	VALUE	Value submitted (can be different for different submit buttons).

Example:

```
<INPUT NAME="submit" TYPE=SUBMIT VALUE="Order">
<INPUT NAME="submit" TYPE=SUBMIT VALUE="Rush order">
```

Rendering:

Select

SELECT		Start of a selection field.
	MULTIPLE	Allow more than one option to be chosen.
	NAME	Symbolic name.
	SIZE	Number of visible elements. SIZE=1 gives pop-down menu, No. of elements > SIZE >1 gives scrolling list, SIZE=No.of elements gives list.
OPTION		Alternative in a selection field.
	SELECTED	This option is initially selected, defaults to first.
	VALUE	Value returned, defaults to content of element.

Example:

```
Colour:
<SELECT SIZE="3" NAME="Colour">
<OPTION SELECTED>Blue <OPTION>Green
<OPTION>Red           <OPTION>Yellow
<OPTION>Brown          </SELECT>
```

Rendering:

SELECTED above was not necessary, since by default the first option is selected.

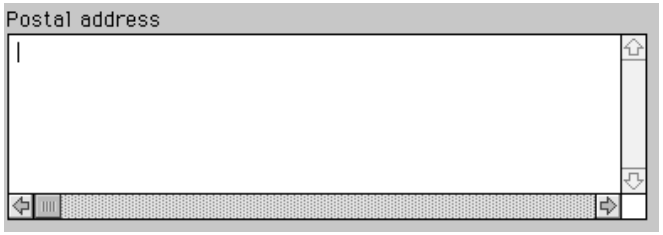
Textarea

TEXTAREA		Multi-line text field.
	COLS	Width in characters of visible field.
	NAME	Symbolic name.
	ROWS	Number of visible rows.

Example:

```
Postal address<TEXTAREA NAME="Address" ROWS="7"
COLS="50"></TEXTAREA><P>
```

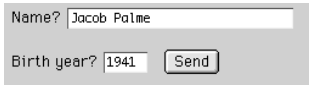
Rendering:



Submission of filled-in forms

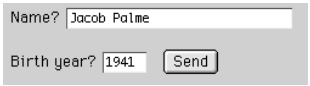
HTTP method	MIME Content-Type	Description
get	application/x-www-form-urlencoded	Very compact single string, appended to the URL
post	multipart/form-data	Volumnious format, every field value becomes its own MIME body part

Example:




```
<FORM action="mailto:foo@bar"
method="POST">
<P>Name? <INPUT TYPE="text" NAME="name"
VALUE="" SIZE=30 MAXLENGTH=30>
<P>Birth year? <INPUT TYPE="text"
NAME="born" VALUE="" SIZE=5
MAXLENGTH=4>  <INPUT
TYPE="submit" NAME="Send" VALUE="Send">
</FORM>
```


Example:




```
<FORM action="mailto:foo@bar" method="POS
<P>Name? <INPUT TYPE="text" NAME="name"
VALUE="" SIZE=30 MAXLENGTH=30>
<P>Birth year? <INPUT TYPE="text"
NAME="born" VALUE="" SIZE=5
MAXLENGTH=4>  <INPUT TYPE="subm
NAME="Send" VALUE="Send"> </FORM>
```

This example will be sent in the following format:

```
From: Jacob Palme <jpalme@dsv.su.se>
MIME-Version: 1.0
To: foo@bar
Subject: ...
Content-type: application/x-www-form-urlencoded

name=Jacob+Palme&born=1941&Send=Send
```

If the first line had been

```
<FORM action="http://www.dsv.su.se/cgi-bin/foo" method="GET">
```

it would have been sent as

```
GET /cgi-bin/foo?name=Jacob+Palme&born=1941&Send=Send HTTP/1.1
```

If the first line had been

```
<FORM action="mailto:foo@bar" method="POST"
enctype="multipart/form-data">
```

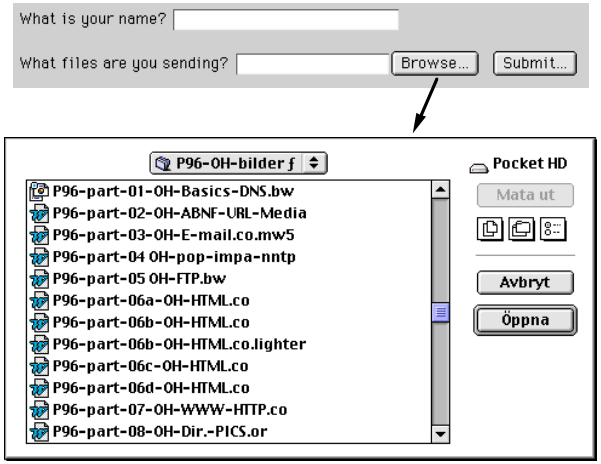
Then it would have been sent as

```
From: Jacob Palme <jpalme@dsv.su.se>
MIME-Version: 1.0
To: foo@bar
Subject: ...
Content-type: multipart/form-data; boundary=++218421377

--++218421377911030
Content-Disposition: form-data; name="name"

Jacob Palme
--++218421377911030
Content-Disposition: form-data; name="born"

1941
--++218421377911030
Content-Disposition: form-data; name="Send"

Send
--++218421377911030--
```

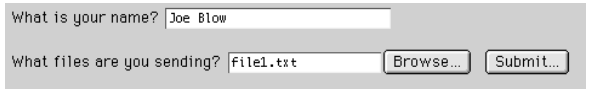
A separate MIME body part for each field value

Form-based File Upload in HTML

(RFC 1867 and HTML 4 chapter 17.13.4 Form Content types)



Form-based File Upload in HTML




```
<FORM ACTION="http://server.dom/cgi/handle"
      ENCTYPE="multipart/form-data"
      METHOD=POST>
What is your name? <INPUT TYPE=TEXT NAME=submitter>
What files are you sending? <INPUT TYPE=FILE NAME=pics>
</FORM>
```

`TYPE=FILE` indicates that a file is requested. `NAME` is to be used when sending the file to indicate which field in the form it applies to. The `ACCEPT` attribute can constrain which file patterns are allowed. The `SIZE` attribute can indicate a size of a field where the files the user has selected are listed. The `VALUE` attribute can be used to give a default file name.

The client might send back the following data:

```
Content-type: multipart/form-data, boundary=AaB03x

--AaB03x
content-disposition: form-data; name="field1"

Joe Blow
--AaB03x
content-disposition: form-data; name="pics"

Content-type: multipart/mixed, boundary=BbC04y

--BbC04y
Content-disposition: attachment; filename="file1.txt"
Content-Type: text/plain

... contents of file1.txt ...
--BbC04y

Content-disposition: attachment; filename="file2.gif"
Content-type: image/gif
Content-Transfer-Encoding: binary

  ...contents of file2.gif...
--BbC04y--
--AaB03x--
```

Note that each field value is sent as a separate MIME body part

Note: Binary data must be encoded

Building applications based on HTTP

Using the form facility of HTML, it is possible to build application programs based on HTML and HTTP. Such an application program uses a web browser as client, and a specially configured HTTP server as server. Almost any computer application which does not require very fast interaction with the user can be built in this way.

+ User does not have to install special client software in his personal computer/workstation.
+ HTML makes it easy to design the user interface.
+ Users may find the web browser interface easy to use because they are accustomed to it.
+ The web browser provides additional facilities automatically, in particular that any page can be printed or saved on a file, and the *Back* and *Forward* buttons in the web browser.

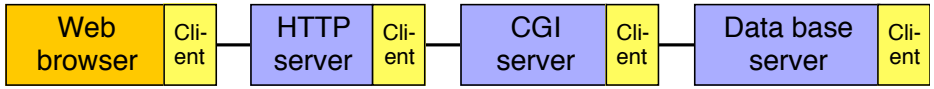
– Sometimes less neat user interface than with a custom-built client.
– Response times sometimes less good than with a custom-built client.
– Applications which require a data base in the client computer cannot be built in this way.

CGI = Common Gateway Interface

CGI is a standard for the interaction between an HTTP server and a special program. CGI allows the HTTP server to recognize special input from the user, for example filled-in-forms, and giving them to an application program. This program can then return a custom-built HTML page to be sent back to the user.

CGI is not the only possible way of doing this. HTTP servers and application can communicate using other methods also.

More about CGI will be said in a special lecture in this course, given by Fredrik Kilander.



Long response times

HTML/HTTP applications which require server knowledge of previous interactions

Many application program requires that information is kept between interactions between a user and the server.

Examples:

- A user logs in, gives his name and password, and can then perform multiple interactions with the server without having to give his name and password again.
- A user retrieves data, and then in a later interaction wants to perform some action based on the data retrieved in the previous interaction.
- A user inputs data, and then in a later interaction wants to perform some action on the data already input (for example change words in a text, add recipients to a message, etc.)
- Suppose for example that we designed software for a user to communicate with his bank. The user might first move some money from one account to another, and then use the moved money to pay a bill.

HTTP (version 1.0) is a stateless protocol

HTTP 1.0 is a stateless protocol. There is no knowledge of previous interactions in the protocol. Every request creates a new interaction, which opens a connection, performs the interaction (for example retrieving data, och sending in a form which the user has filled in). After data has been transmitted, the HTTP connection between the client and the server is broken. Thus, HTTP 1.0 as such is not suitable for sessions of multiple interactions between user and server, unless some special trick is used. Also HTTP 1.1 is inpractice mostly used as a stateless protocol.

Here are some such special tricks:

(1) Store session information in custom-built URLs.

(2) Store session information in hidden fields in a form.

(3) Use cookies.

(1) Store session information in custom-built URLs.

When the server creates the custom-built web page to be sent to the user, the server can store session information in specially built URLs. When the user clicks on these URLs, this in formation is sent back to the server.

Example: The server creates a URL like this:

```
HTTP:/www.dsv.su.se/exam-results?per-nils+sf14ty
```

where `per-nils` is the user account and `sf14ty` is the user password.

It is somewhat dangerous to store passwords in URL-s which other people might see and use. To reduce this risk, often a special session password is used, with more limited applicability. For example, the session password will become invalid if there is no interaction in 10 minutes. Such session passwords are often named *Magic cookie*. A *Magic cookie* is a special password which gives the user some special rights, often only at a certain time. A *Magic cookie* often also gives the server information to identify the user, so that it can replace both the user name and the user password.

(2) Store session information in hidden fields in a form.

If the interaction is made by the server sending forms to the user, and the user returning the filled in forms, then the server can store session information in the forms sent to the user. This can be done in open fields, if the user needs to see the information sent, or in hidden form fields, if the user would not want to see it. The contents of the hidden fields are sent back with the filled-in form to the server. Two common usage of such hidden fields are:

(a) To store user account names, passwords or magic cookies. This information will help the server to look up the user information.

Disadvantage: Server has to store information about each concurrent user.

(b) To store full information of what the user has achieved. For example, an application where the user interactively creates a budget, the whole budget could be sent back in each interaction. Thus, the server need not remember anything of previous interactions, all of it is provided in data sent to the server with every interaction.

Disadvantage: The amount of information sent back and forward between user and server must not get too large, or the response times will be less good. If, for example, a 28800 bps connection is used and a maximum delay of 5 seconds is acceptable, then a maximum of 28800*5/2*10 = 72 Kbytes is acceptable.

(3) Use cookies

Newer browsers have a *cookie* facility, with which a server can store a “cookie”, i.e. some kind of session-ID, in the web browser, which the server at a later time can query the value of.

(More about cookies in the HTTP lecture.)

Version 1.1 of HTTP has facilities to support *persistent connections.*

The disadvantage with these methods, is that they are not supported by all web browsers, and that some users set their browsers to not accept cookies, because they believe cookies to be an infringement of their privacy.

HTML tables

A feature in HTML 3.2, which is much used and supported by many browsers, is HTML tables.

Table example 1:

This example can be found at URL:

```
HTTP://www.dsv.su.se/~jpalme/test/table-example-1.html
```

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

HTML code behind table example 1:

```
<TABLE>
<CAPTION>Calendar for September 1996</CAPTION>
<TR><TH>Sunday</TH><TH>Monday</TH><TH>Tuesday</TH>
<TH>Wednesday</TH><TH>Thursday</TH><TH>Friday</TH>
<TH>Saturday<BR></TH></TR>
<TR><TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD><TD>5</TD>
<TD>6</TD><TD>7<BR></TR>
<TR><TD>8</TD><TD>9</TD><TD>10</TD><TD>11</TD><TD>12</TD>
<TD>13</TD><TD>14<BR></TR>
<TR><TD>15</TD><TD>16</TD><TD>17</TD><TD>18</TD><TD>19</TD>
<TD>20</TD><TD>21<BR></TR>
<TR><TD>22</TD><TD>23</TD><TD>24</TD><TD>25</TD><TD>26</TD>
<TD>27</TD><TD>28<BR></TR>
<TR><TD>29</TD><TD>30</TD></TR>
</TABLE>
<BR CLEAR=LEFT>
```

Column width autofit is very useful for tables with lots of

Merging cells

It is possible to merge adjacent cells both horizontally and vertically into arbitrary rectangular shapes.

A test table with merged cells

Sex	Body measures		Hair colour	Payment Status
	height	weight		
John	185	75	Brown	Paid
Mary	175	65	Red	Paid

```
<TABLE BORDER=1>
  <CAPTION>A test table with merged cells</CAPTION>
  <TR><TH ROWSPAN=2>Sex<TH COLSPAN=2>Body measures
      <TH ROWSPAN=2>Hair<BR>colour<TH
ROWSPAN=2>Payment<BR>Status
  <TR><TH>height<TH>weight
  <TR><TH>John<TD>185<TD>75<TD>Brown<TD>Paid
  <TR><TH>Mary<TD>175<TD>65<TD>Red<TD>Paid
</TABLE>
```


Text flowing around a table

A test table with merged cells

Sex	Body measures		Hair colour	Payment Status
	height	weight		
John	185	75	Brown	Paid
Mary	175	65	Red	Paid

If a table is narrow, and if you specify <TABLE BORDER ALIGN=LEFT>, the following text will flow around the table, as is shown by this example. If this is not what you want, put <BR CLEAR=LEFT> immediately after the end of the table. The statement <BR CLEAR=LEFT> instructs the web browser to statement after the end of the table and not at the side of the table. Text after the end of the table

HTML markup for text flowing around a table

```
<TABLE BORDER=1 ALIGN=LEFT>
  <CAPTION>A test table with merged cells</CAPTION>
  <TR><TH ROWSPAN=2>Sex<TH COLSPAN=2>Body measures
      <TH ROWSPAN=2>Hair<BR>colour<TH
ROWSPAN=2>Payment<BR>Status
  <TR><TH>height<TH>weight
  <TR><TH>John<TD>185<TD>75<TD>Brown<TD>Paid
  <TR><TH>Mary<TD>175<TD>65<TD>Red<TD>Paid
</TABLE>
```

```
If a table is narrow, and if you specify &lt;TABLE BORDER
ALIGN=LEFT&gt;, the following text will flow around the table,
as is shown by this example. If this is not what you want, put
&lt;BR CLEAR=LEFT&gt; immediately after the end of the table.
The statement &lt;BR CLEAR=LEFT&gt; instructs the web browser
to statement after the end of the table and not at the side of
the table.
```

To cause text to start below the table and not flow around it, put the element **<BR CLEAR=LEFT>** before the text, or do not put **ALIGN=LEFT** in the **<TABLE>** element.

Java, Javascript (ECMAScript)

Program (s.k. applets) som laddas ner från nätet samtidigt med websidor. Kan köras så snart de laddats ner. Exempel på användningar:

- Minska svarstiden genom lokal interaktion istället för server-interaktion
- Rörliga bilder och andra saker som inte så enkelt kan göras med vanlig HTML
- Begränsade av säkerhetsproblem

Skillnaden mellan Java och Javascript

Egenskap	Java	Javascript
Lagring	I separata filer, till vilka det finns länkar i HTML-texten.	Som del av HTML-texten.
Kompilering	Kompileras till bytekod, som sedan interpreteras, eller till objetkod som exekveras.	Källkoden interpreteras direkt.
Funktioner	Generellt programmeringsspråk. observera dock säkerhetsbegränsningarna.	Mest kommandon för att styra webläsaren. Kan idag inte ändra i redan visad web-sida (däremot skriva nya sidor), detta kommer snart att ändras!

*:96 Overheads

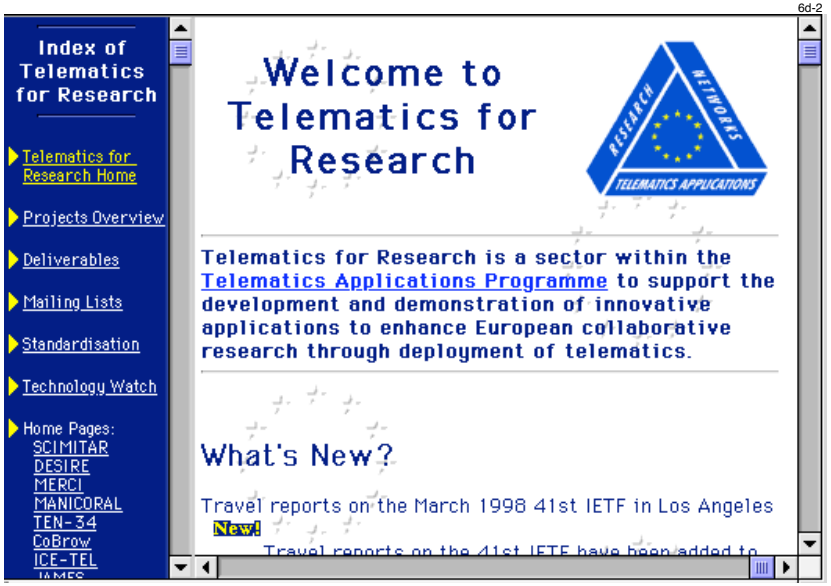
Part 6d: HTML and CSS (Frames, HTML 4.0, Good HTML, Testing HTML)

More about this course about Internet application protocols can be found at URL:

http://www.dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html

Last update: 04-09-05 13.32

FRAMES




```
<HTML>
<HEAD>
<TITLE>Telematics for Research Home</TITLE>
</HEAD>

<FRAMESET COLS="120,*" BORDER="0" FRAMEBORDER="0">
<FRAME SRC="navbar.html" NAME="navbar" MARGINWIDTH="0"
MARGINHEIGHT="0" SCROLLING="auto">
<FRAME SRC="rtdnhome.htm" NAME="body">
</FRAMESET>
<NOFRAMES><BODY>
<P><B>Your browser does not appear to support
frames.</B>
```

Extract from the file navbar.html:

```
<FONT SIZE=-1><A HREF="/" TARGET="_top"><FONT
COLOR="#FFFF00">Telematics for Research Home</FONT></A>
```

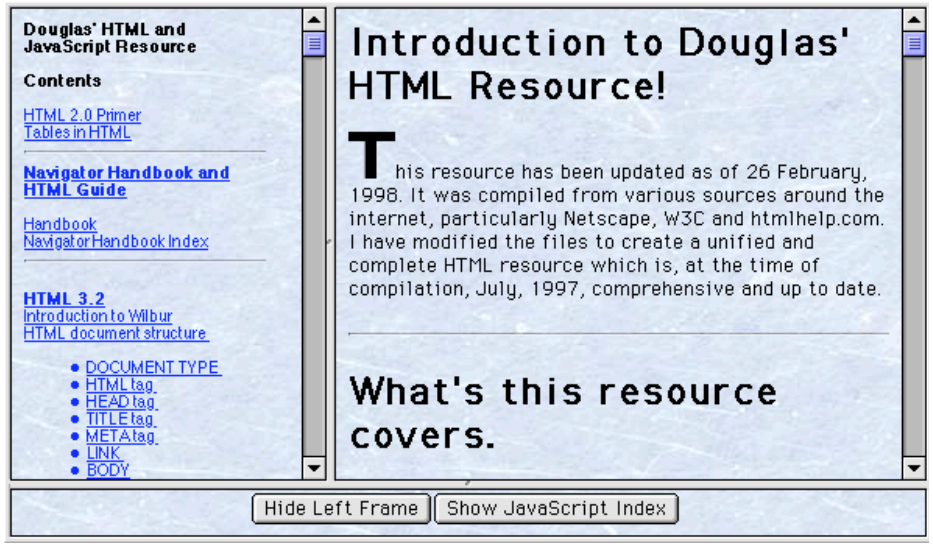
`TARGET="_top"` reloads the whole window,
`TARGET="body"` only the left frame.

Frame target names (6.16 in HTML 4.0 specification)

Except for the reserved names listed below, frame target names must begin with an alphabetic character (a-z, A-Z). User agents should ignore all other target names. The following target names are reserved and have special meanings.

`_blank`	The user agent should load the designated document in a new, unnamed window.
`_self`	The user agent should load the document in the same frame as the element that refers to this target.
`_parent`	The user agent should load the document into the immediate `FRAMESET` parent of the current frame. This value is equivalent to `_self` if the current frame has no parent.
`_top`	The user agent should load the document into the full, original window (thus cancelling all other frames). This value is equivalent to `_self` if the current frame has no parent.

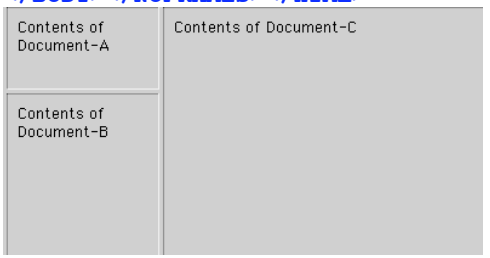
http://www.ozemail.com.au/~phoenix1/html/index.htm



Two frames on top of each other, the top frame contains two subframes, left and right.

Frame notation

```
<HTML><HEAD></HEAD>
<FRAMESET   COLS = "20%,80% "  >
   <FRAMESET  ROWS = "100,200 "  >
      <FRAME SRC="Document-A" NORESIZE>
      <FRAME SRC="Document-B" NORESIZE>
   </FRAMESET>
   <FRAME SRC="Document-C" RESIZE>
</FRAMESET>
<NOFRAMES>
<BODY>Text for non-frame readers
</BODY></NOFRAMES></HTML>
```



cols = horizontal

rows = vertical

"20%,80%" = percentage of window

"100,200" = pixels

"3*,1*" = relative sizes

Frame element attributes

name = *cdata*	Name, used in TARGETs.
longdesc =	Link to a long description of the frame for non-visual user agents.
src = *uri*	Location of the initial contents to be contained in the frame.
noresize	Tells the user agent that the frame window must not be resizeable.
scrolling = **auto|yes|no**	Scroll information for the frame window. Possible values: **auto:** Scrolling devices for the frame window when necessary. This is the default value. **yes:** Always provide scrolling devices for the frame window. **no:** Do not to provide scrolling devices for the frame window.
frameborder = **1|0**	Possible values: **1:** A separator, default value. **0:** No separator.
marginwidth =	Space to be left between the frame's contents in its left and right margins.
marginheight = *pixels*	Space to be left between the frame's contents in its top and bottom margins.

Problems with Frames

1. Difficult for sight-impaired readers.
2. Difficulty with links and bookmarks:
   a. Search engines may produce link to a frame instead of a page.
   b. Bookmarks, history links, etc.
   c. Other linking problems.
3. Difficulty printing (print page, or print frame content?)

Because of these problems, many people regard frames as not good HTML design practice.

Frames, tables or subwindows?

Frame:	Designer controls more, un-scrolling frame always visibile to user. May allow user reformatting and scrolling of each frame.
Tables:	Whole window scrolls, avoids problems with frames.
Subwindows: **<A HREF=... TARGET=... >**	User can separately reposition and reformat each window, and windows can overlay each other on the screen.

Embedding External Content

<OBJECT ...> element

Example of old notation

```
<EMBED width=150 height=200 SRC="my-movie.dcr">
```

Example of new notation

```
<OBJECT data=my-movie.dcr
    type="application/director"
    width=150 height=200>
<IMG src=use-schockwave.gif alt="Get schockwave">
</OBJECT>
```

The **<IMG ...>** above is only shown to users of browsers which do not understand the **<OBJECT ...>** notation.

Two Layout Methods:

Using tables	Works with all browsers.
Using CSS	More features, better for sight-impaired readers, problems with some variants and some browsers (2004, may disappear in the future).

Use of tables for controlling positioning of text and graphics

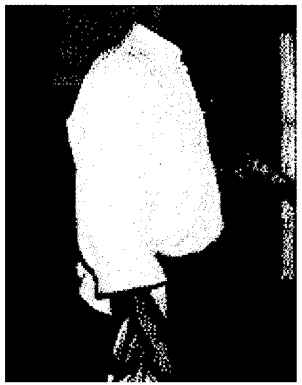
All the news that's fit to print

Headless corpse found on cloudless day

Frightened visitors at Yosemite National Park in California though they had witnessed one of the all too common crimes in the Californian underworld.

However, it was later revealed that the headless corpose was in fact only a wax doll, put there as a prank by som college youngsters.

Apparently, the youngsters did this as an initation rite into a secret society for. Police comment that pranks all too often cause maybe unintented misery to frightened onlookers. The district attorney says he is going to prosecute to the extent that the law permits



This is how the corpse looked like to the onlookers.

Short news

The old city Two Rothweiler dogs fought below the castle. Panic was close, but their owners succeeded in getting them separated. A tax was believed to be the cause of the fight.

Water festival There are twenty ladies in queue in front of the insufficient toilets. "A disgrace", one of them said.

Kaknäs tower The famous American country and western singer John Rohrstadt performed in the

Another example of the use of tables

March 96 EWOS/ETSI EG-MHS meeting

March 26-28, 1996, Brussels

It is now becoming clear that the MHS group will cease to exist in its current form after the re-structuring of EWOS, since the level of activities in the group has diminished ...

READ NOW

Succesful NameFLOW - PARADISE meeting during EEMA '96

11 June 1996, Brussels

The 4th NameFLOW - PARADISE (NP) punters meeting took place during the first day of the EEMA '96 conference in Brussels. The meeting was short, only half a day, but was rather successful...

STEP meeting during JENC7, the presentation of standardization to the Central and Eastern European countries

May 15th 1996, Budapest (Hungary)

This third STEP Project Meeting was hosted in Budapest by MSZT, the Hungarian Standards Institution, on 16-17 May. As the 7th Joint European Networking Conference was also taking place in Budapest at the same time, an ad-hoc STEP meeting had been planned in the morning...

READ NOW

June 96 EWOS/ETSI EG-MHS & EG-DIR meetings

June 24-27, 1996, Brussels

As the activities within the MHS group have now

How this was done (Mozilla HTML!!):

```
<TABLE> <TR VALIGN=TOP><TD WIDTH="45%">
<CENTER><P><B><FONT COLOR="#004080"><FONT SIZE=+1>
March 96 EWOS/ETSI EG-MHS meeting
</FONT></FONT></B></P></CENTER>

<CENTER><P><I>March 26-28, 1996, Brussels</I></P></CENTER>

<P><FONT SIZE=-1>It is now becoming clear that the MHS group
will cease to exist in its current form after the re-
structuring of EWOS, since the level of activities in the
group has diminished ...</FONT></P>

<CENTER><P><A HREF="http://icl1.iihe.ac.be:8080/internal-
report/stc-96-08.html">
<IMG SRC="read_it.gif" BORDER=0 HEIGHT=20 WIDTH=139
ALIGN=CENTER></A></P></CENTER>

<CENTER><P><HR NOSHADE WIDTH="100%"><BR>
 ... ... ...
<TD ALIGN=CENTER VALIGN=TOP WIDTH="10%"><P> <IMG
SRC="Vertical.GIF" HEIGHT=800 WIDTH=3 ALIGN=ABSCENTER></P>
</TD>
 ... ... ...
<TD WIDTH="45%">
```

Warning: **<CENTER>** is maybe Netscape specific? The correct HTML standard form for this is **<P ALIGN=CENTER>**.

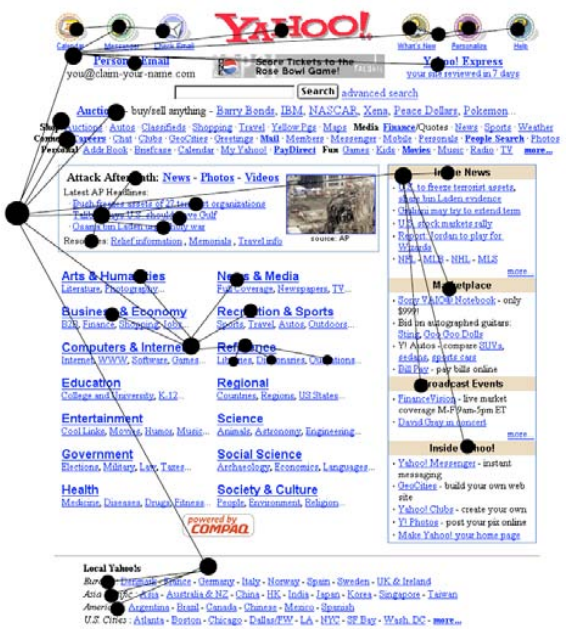
<DIV ...> and <SPAN ...> elements

Three similar examples:

`<CENTER>This text is centered.</CENTER>`	Non-standard.
`<DIV ALIGN=CENTER>This text is centered.</DIV>`	Works best with new browsers.
`<P ALIGN=CENTER>This text is centered.</P>`	Similar to above, but larger line distance.

<SPAN ...> is similar to <DIV...> but does not enforce a new line. Thus <SPAN ...> cannot be used together with attributes which can only be applied to a whole line, like ALIGN=CENTER in the example above.

Making Web Pages Suitable for Disabled People: Make implicit structure explicit



Making Web Pages Suitable for Disabled People

Implicit structure can be supported by:

```
<H1>, <H2>, etc. tags in logical order
<DIV title="department 1">
  <DIV title="department 1.2">
  </DIV
</DIV>
```

All graphics must be labelled, for example:

```
<IMG src="sweden.gif" alt="Map of Sweden">
```

Sometimes reference to an external document is better:

```
<IMG src="sitemap.gif" alt="Site map"
longdesc="sitemap-textual.html">
```


Warnings

- Do not assume that all readers are using a particular browser, or even graphical browsers. Also remember that many users set their browsers to not download graphics automatically.
- Remember that some readers use portable computers with small screens. Avoid layout which requires screens larger than 14 inches (640x480 pixels, 22x17 cm).
- Avoid other than very light backgrounds. Backgrounds may not look on all computers as they look on your screen, and often you see in the web pages with text which is difficult to read because of a too strong background.

HTML testing

Most web browsers are very permissive, and accept a lot of incorrect HTML. But exactly what incorrect HTML is accepted, and how this is handled, varies between web browsers. This means that if you test your HTML markup with one web browser, it may still contain faults which will result in very unacceptable results with other web browsers.

To avoid this, you can test your web documents.

For access to HTML 4.0 test suites, look at URL

```
http://validator.w3.org/
```

Weblint, Tidy

Weblint is a computer program which will scan an HTML markup file and find faulty HTML markup. It is available for most major platforms.

Tidy is another program, which also reformats the HTML code for better readability (when reading the code, user view is not changed)

How to indicate which HTML version you are using

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
"http://www.w3.org/TR/REC-html40/frameset.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```


Example of part of a test report received from an on-line test facility:

The text below is an extract from the test report from the on-line test facility at URL:
`http://ugweb.cs.ualberta.ca/~gerald/validate/`

```
Error at line 96:
<H2><A NAME="chapter1">Title of chapter 1</H2>


end tag for `A' omitted, but its declaration does
not permit this (explanation...>
```

Sorry, this document does not validate as HTML.

Writing good HTML

Many Web-page designers live in a fantasy world. In that world, everyone has very fast computers, very large screens and T1 access. We recently met an executive responsible for the overall design of a Web site for a major entertainment company. The exec spoke in glowing terms of the "success" of the site.

One evening we decided to visit the page through one of our Web access points. The page took nine minutes to appear on a 28,800-baud modem. Most designers still fail to realize that there is a 1 0-million person audience out there that will see their work at a much slower rate than the one at which it was developed. We would like to suggest that developers consider dual sites-one for the fast lane and another for those of us who travel the speed limit.

Some people actually use the Web to access information, not to see fancy graphics or jumping Java scripts. Remember: the Internet started out as a way to distribute information effectively in the event of a nuclear attack. In the process of getting to the pertinent information on most Web sites today, you would be dead before the wallpaper loads.

There are ways to create a well-designed and appealing site without loading it down with bells and whistles. Such tricks of the trade have everything to do with understanding, designing, and creating specifically for the Web as if it were a unique medium, which it is.

Daniel Lorenzetti and Linda Rice Lorenzetti in "The Good, the Bad and the Ugly" in OnTheInternet, July/August 1996.

*:96 Overheads

Part 6e: Cascading Style Sheets (CSS)

More about this course about Internet application protocols can be found at URL:

http://www.dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html

Last update: 05-01-14 19.24

Rendering:

Here is one **word**, which is boldfaced and with a smaller font.

Old notation:

```
Here is one <B><FONT SIZE="2">word</B></FONT> which is
boldfaced and with a smaller font.
```

New notation:

```
<HEAD>
<STYLE type="text/css">
SPAN.xyz { font-size:10pt; font-style: bold }
</STYLE>
</HEAD><BODY>
Here is one <SPAN class="xyz">word</SPAN> which is boldfaced
and with a smaller font.
Note: { } surrounds a group of properties for a certain
style.
```


Why is CSS good?

CSS separates formatting from logical content, which has many advantages,

1. formatting codes need not be repeated,
2. it is easier to produce text if it is separated from formatting mark-up,
3. the risk of errors is probably smaller,
4. it is easier to format the same content in multiple formats,
5. it is easier to modify the formatting for many pages,
6. tasks can be better separated between content producer and format designer.

Relative and absolute font sizes

Before we got style sheets, HTML only allowed relative font sizes. A user could specify <font size="n"> where the following values of "n" were allowed:

Non-signed value	Signed value	Example of possible rendering, for a Windows user with 12 pt as default font size using the Arial font (50 % larger here)
1	-2	The quick brown fox
2	-1	The quick brown fox
3	0	The quick brown fox
4	+1	The quick brown fox
5	+2	The quick brown fox
6	+3	The quick brown fox
7	+4	The quick brown

With CSS, absolute fonts became possible

With CSS, absolute fonts became possible, for example

```
<style type="text/css"> <!--
.arial8 {  font-family: Arial; font-size: 8pt}
--> </style>
```

Absolute font size	Windows rendering (96 pixels/inch)	Macintosh rendering (72 pixels/inch)
8 pt	Arial 8 pt	Arial 8 pt
10 pt	Arial 10 pt	Arial 10 pt
12 pt	Arial 12 pt	Arial 12 pt

Note: The pictures have been increased 50 % (with corresponding reduced resolution) to become more readable on an overhead screen.

Command links should not change colour

Below, a command bar is produced using a HTML table (saves download time compared to using graphical buttons). But the links in the command bar should not change colour when they have been visited, like the links in the text below the command bar.

Visited and unvisited links have the same colour:

Hem	Personligt	Logga ut	Web4Groups	Skriva brev	Hjälp

Visited and unvisited links have different colour:

Hem	Personligt	Logga ut	Web4Groups	Skriva brev	Hjälp

Archives of this list are available by anonymous ftp from FTP://SEGATE.SUNET.SE. You can also browse the archives by http from HTTP://segate.sunet.se/archives/mailnews-1.html. The FTP archives are better if you want to download all messages, the HTTP archives are better if you want to browse and find a particular message only.

How to stop links changing colour when visited?

Old method: only works for all links in the whole document:

```
<BODY BGCOLOR="#FFFFFF" LINK="#003399" VLINK="#003399"
ALINK="#003399">
```

Using style sheets (the HTML below is abbreviated):

```
<HTML><HEAD>
    <TITLE>CSS and link colour</TITLE>
    <STYLE type="text/css">
    A.allblue:visited { color: #003399 }
    A.allblue:link { color: #003399 }
</STYLE></HEAD><BODY BGCOLOR="#FFFFFF">
<FONT SIZE=5>Visited and unvisited links have the same colour:
<TABLE BORDER="1" CELLPADDING="4" CELLSPACING="1" WIDTH="500">
  <TR><TD BGCOLOR="#FFCC00">
    <A class="allblue" HREF="hem.html"
    target="subwindow">
    <FONT FACE="Geneva, Helvetica" SIZE=3>
    <B>Hem</A></TD></TABLE>
<FONT SIZE=5>Archives of this list are available from
<A HREF="//SEGATE.SUNET.SE/">FTP://SEGATE.SUNET.SE
</A>.You can also browse the archives by http from
```

This link will not change colour

This link will change colour

Using CSS to avoid too long rows

For good readability, text rows should contain about 40-60 characters per line. But if the user has set his browser to a wide window (which the user needs when viewing other web pages) ordinary text will be too wide.

Archives of this list are available by anonymous ftp from FTP://SEGATE.SUNET.SE. You can also browse the archives by http from HTTP://segate.sunet.se/archives/mailnews-1.html. The FTP archives are better if you want to download all messages, the HTTP archives are better if you want to browse and find a particular message only.

Archives of this list are available by anonymous ftp from FTP://SEGATE.SUNET.SE. You can also browse the archives by http from HTTP://segate.sunet.se/archives/mailnews-1.html. The FTP archives are better if you want to download all messages, the HTTP archives are better if you want to browse and find a particular message only.

Archives of this list are available by anonymous ftp from FTP://SEGATE.SUNET.SE. You can also browse the archives by http from HTTP://segate.sunet.se/archives/mailnews-1.html. The FTP archives are better if you want to download all messages, the HTTP archives are better if you want to browse and find a particular message only.

This shows Netscape rendering of text shows the same text in three ways:

1. Ordinary plain text with a wide browser window
2. Using a table with `<TD WIDTH=400>`
3. Using style sheets with `<P class=`

Using CSS to avoid too long rows

```
<HTML><HEAD>
  <TITLE>CSS and link colour</TITLE>
  <STYLE type="text/css">
  P.narrow { width: 400 }
  </STYLE>
</HEAD><BODY BGCOLOR="#FFFFFF">
```

Table method

```
<TABLE BORDER="0" CELLPADDING="8" CELLSPACING="0"
WIDTH="400"><TR><TD>
Iamque fretum Minyae Pagasaea puppe secabant,
perpetuaque trahens inopem sub nocte senectam.
</TD></TR></TABLE>
```

CSS method

```
<P class="narrow">
Iamque fretum Minyae Pagasaea puppe secabant,
perpetuaque trahens inopem sub nocte senectam.</p>
```


Using style sheets for absolute positioning

(may not work with all browsers)

Department of Computer and Systems Sciences

Back

Save on Disk

A flower to CSS

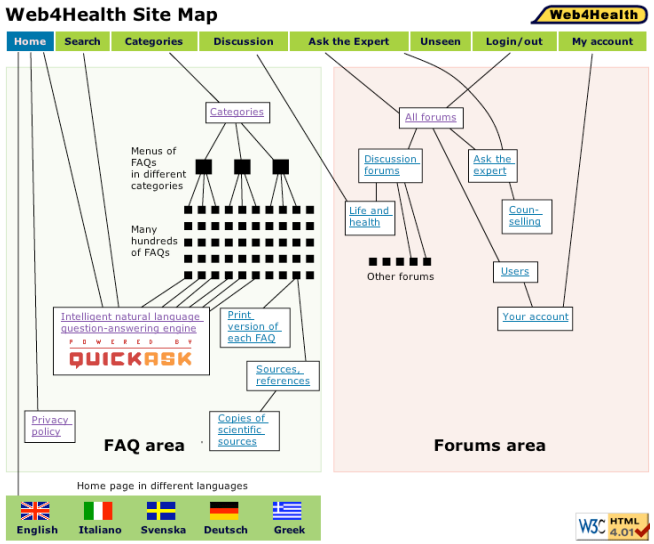
Using style sheets for absolute positioning

```
<HTML><HEAD><TITLE>CSS position command</TITLE>
    <STYLE type="text/css">
    div.dsvtext {position: absolute; left: 135px; top: 10px;
    right:auto; bottom: auto; width:150 }
    div.backimg {position: absolute; left: 300px; top: 60 px;
    right:auto; bottom: auto }
    div.backtext {position: absolute; left: 305px; top: 95 px;
    width:100 px; bottom: auto }
    div.saveimg {position: absolute; left: 40px; top: 130px;
    right:auto; bottom: auto }
    div.savetext {position: absolute; left: 20px; top: 130 px;
    width:100 px; bottom: auto }
    div.flowerimg {position: absolute; left: 150px; top:
    150px; right:auto; bottom: auto }
    </STYLE>
</HEAD><BODY BGCOLOR="#FFFFFF">

<IMG SRC="DSV-logo123x57.gif" WIDTH="123" HEIGHT="57"
ALIGN="BOTTOM" BORDER="0" ALT="DSV-logo">
<DIV class="dsvtext"><FONT SIZE=5>Department of
Computer and Systems Sciences</FONT></DIV>
```

continued


```
<DIV class="backimg"><IMG SRC="Arrow.GIF" WIDTH="45"
HEIGHT="31" ALIGN="BOTTOM" BORDER="0" ALT=" &lt;= "></DIV>
<DIV class="backtext"><FONT SIZE=5>Back</FONT></DIV>

<DIV class="saveimg"><IMG SRC="disk.gif" WIDTH="32"
HEIGHT="32" ALIGN="BOTTOM" BORDER="0" ALT="Disk"></DIV>
<DIV class="savetext"><FONT SIZE=5>Save on Disk</FONT></DIV>

<DIV class="flowerimg"><IMG SRC="flower1.gif" WIDTH="28"
HEIGHT="29" ALIGN="BOTTOM" BORDER="0" ALT="Flower">
<FONT SIZE=5>A flower to CSS</FONT></DIV>

</BODY></HTML>
```

Web page with absolute positioning: Site map



Advantage: Same background and formatting can easily be used to produce this web page in multiple languages (English, Swedish, etc.)

Extract of CSS used in the site-map example:

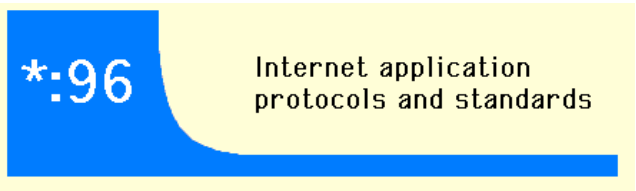
```
.text {
font-family: verdana,geneva,helvetica,arial,non-serif;
font-size: 11px
}
#categories { position: fixed; left: 241px; top: 123px
}
```

Extract of HTML used in the site-map example:

```
<div class="text" id="categories">
   <a href=
   "/en/answers/project-all-menus.htm">Categories</a>
</div>
```

Note: “class” is suitable for formatting which is used many times, “id” for formatting used only once.

Using tables for exact positioning




```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0>
    <TR><TD ROWSPAN=2 BGCOLOR="#0066FF">
            <CENTER>
            <FONT SIZE=7 COLOR="#FFFFFF"><B>*:96</B></FONT>
             <BR> </TD>
        <TD WIDTH=73 HEIGHT=100 BGCOLOR="#0066FF">
            <IMG SRC="exam-98-05a.gif" WIDTH=73 HEIGHT=100>
        </TD>
        <TD WIDTH=254 HEIGHT=100 BGCOLOR="#FFFFCC">
            <FONT SIZE=5><B>Internet application<BR>
            protocols and standards</B></FONT>
        </TD>
    </TR><TR>
        <TD HEIGHT=14 COLSPAN=2 BGCOLOR="#3366FF"> </TD>
    </TR>
</TABLE>
```


Selectors in CSS

.big { font-size: 24px }	<p class=big>,
p.big { font-size: 24px }	<p class=big>
div.big { font-size: 24px }	
p#big { font-size: 24px }	<p id=big>
a.dynamic:hover { color: red ; font-weight: bold}	<a class=dynamic ...
a.dynamic:link { color: blue ; font-weight: normal}	
a.dynamic:visited { color: red ; font-weight: normal}	
h1 em {color: red }	This headline is very important

There are many more selector variants in the CSS recommendation.

Cascading in Cascading Style Sheets

Style sheets may be specified by

(a) Author

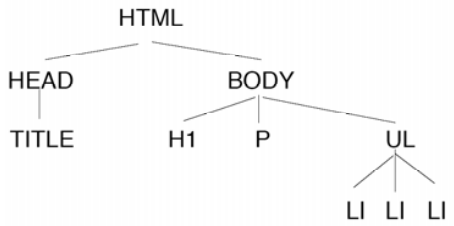
(b) Reader, specially for this document

(c) Defaults in the browser, set at delivery or modified by the user

How, then, should conflicting style sheet information be combined?

Cascading in Cascading Style Sheets

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML><HEAD>
<TITLE>My home page</TITLE>
</HEAD><BODY>
<H1>My home page</H1>
<P>Welcome to my home page! Let me tell you about my favorite
composers:
<UL>
<LI> Elvis Costello
<LI> Johannes Brahms
<LI> Georges Brassens
</UL>
</BODY>
</HTML>
```



Measurements and inheritance:

Example of use: `<p style="font-size:12px"> ... <em style="font-size: 0.8ex">...</p>`

cm, mm, in		`h1 {padding-top: 5mm }`
pixels[1]	px	`h1 { font-size: 12px }`
points = inch/72	pt	`p { word-spacing: 20pt }`
relative to inherited size	em	`body {font-size: 16px }` `h1 {margin: 0.5em }`
	ex	`h1 {margin-left: 0.5ex }`
	%	`h1 {line-height: 120% }`

[1] Converted when printed on paper based on 96 pixels/inch.

Cascading order (section 6.3.1 in the CSS recommendation)

To find the value for an element/property combination, user agents must apply the following algorithm:

1. Find all declarations that apply to the element/property in question. Declarations apply if the associated selector matches [p. 43] the element in question. If no declarations apply, terminate the algorithm.
2. Sort the declarations by explicit weight: declarations marked '!important' carry more weight than unmarked (normal) declarations. See the section on 'important' [p. 60] rules for more information.
3. **Sort by origin: the author's style sheets override the user's style sheets which override the default style sheet.** An imported style sheet has the same origin as the style sheet from which it is imported.
4. Sort by specificity of selector: more specific selectors will override more general ones. The definition and calculation of specificity is object-language dependent. Pseudo-elements and pseudo-classes are counted as normal elements and classes, respectively.
5. Sort by order specified: if two rules have the same weight, the latter specified wins. Rules in imported style sheets are considered to be before any rules in the style sheet itself.

The search for the property value must be terminated when any of the above steps yields a rule that has a higher weight than the other rules that apply to the same element/property combination.
This strategy gives author's style sheets considerably higher weight than those of the reader. It is therefore important that the User agent gives the user the ability to turn off the influence of a certain style sheet, e.g., through a pull-down menu.

Linking to external style sheet

HTML example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <link href="http://web4health.info/web4health-v2.css"
   rel="stylesheet" type="text/css">
   <link href="http://web4health.info/site-map.css"
   rel="stylesheet"
   type="text/css">
</head>
```

XML example:

```
<?xml-stylesheet type="text/css" href="bach.css"?>
<ARTICLE>
  <HEADLINE>Fredrick the Great meets Bach</HEADLINE>
  <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
  <PARA>
  ... ...
```


Shorthand properties in CSS

```
h1 {
font-weight: bold;
font-size: 12pt;
line-height: 14pt;
font-family: Helvetica;
font-variant: normal;
font-style: normal;
}
```

may be rewritten with a single shorthand property:

```
h1 { font: bold 12pt/14pt Helvetica }
```

In this example, 'font-variant', and 'font-style' take their initial values.

HTML STYLE Attribute

Instead of

```
<HEAD>
<STYLE type="text/css">
SPAN.xyz { font-size:10pt; font-style: bold }
</STYLE>
</HEAD><BODY>
Here is one <SPAN class="xyz">word</SPAN> which is boldfaced and with
a smaller font.
```

One can write

```
Here is one <SPAN style="font-size:10pt; font-style:
bold">word</SPAN> which is boldfaced and with a smaller font.
```

Disadvantage: Eliminates many of the advantages of separate style sheets.

Advantage: Easier for minor or automatic adaption of existing HTML code. Can use CSS commands in HTML.

Example of use: Google cached search results for the query "Olof Palme murder" using style attribute:

On February 28, 1986 Swedish Prime Minister Olof Palme was gunned down on a Stockholm street, as he was walking home from a cinema with his wife. While there are various theories about who could have been behind the murder, the identity of the culprit remains a mystery.

Special formatting of *first line* of a paragraph

CSS

```
p:first-line {text-transform:
uppercase }
```

HTML

```
<p>Iamque fretum Minyae Pagasaea
puppe secabant, perpetuaque trahens
inopem sub nocte senectam Phineus
visus erat, iuvenesque Aquilone
creati virgineas volucres miseri
senis ore fugarant.</p>
```

Rendering

IAMQUE FRETUM MINYAE PAGASAEA puppe secabant, perpetuaque trahens inopem sub nocte senectam Phineus visus erat, iuvenesque Aquilone creati virgineas volucres miseri senis ore fugarant.

Special formatting of *first letter* of a paragraph

CSS
```
p:first-letter {font-size: 3em }
```

HTML
```
<p>Iamque fretum Minyae Pagasaea
puppe secabant, perpetuaque trahens
inopem sub nocte senectam Phineus
visus erat.</p>
```

Rendering

I amque fretum Minyae Pagasaea puppe secabant, perpetuaque trahens inopem sub nocte senectam Phineus visus erat.

Special formatting of *first word* of a paragraph

CSS
```
span.largeword {font-size: 2em ;
text-transform: uppercase }
```

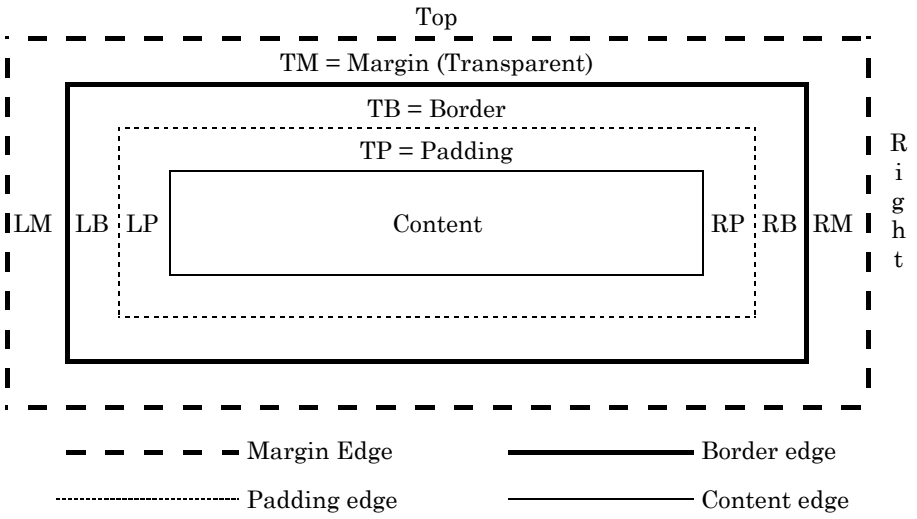
HTML
```
<p><span
class=largeword>Iamque</span> fretum
Minyae Pagasaea puppe secabant,
perpetuaque trahens inopem sub nocte
senectam Phineus visus erat.</p>
```

Rendering IAMQUE fretum Minyae Pagasaea puppe secabant, perpetuaque trahens inopem sub nocte senectam Phineus visus erat.

Different formatting for screen viewing and printing

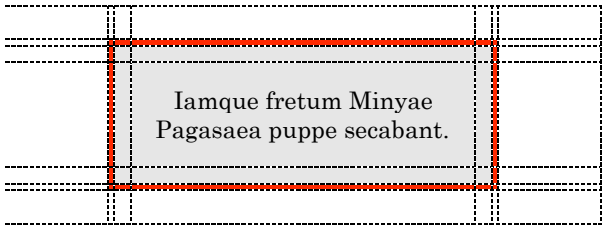
```
@media  print {
.dummy {} /* some faulty browser need this */
.pagebreak
{ page-break-before: always }
h1 { font-size: 18pt }
} /* end of media print */

@media screen, print {
.dummy {}
h1 {   font-family: Verdana, Arial, Helvetica, sans-serif;
       text-align: left;
       padding-top: 0.2cm;
       padding-bottom: 0.2cm;
       font-weight: bold }

} /* end of media screen, print */

@media screen{
.dummy {}
h1 { font-size: 26px }
} /* End of media screen */
```


Box model



Example of use of box model

```
<style type="text/css">
div.mybox {
    margin: 1cm 3cm;   border: 5pt solid red;
    padding: 5mm;      background-color: #CCCCCC;
    text-align: center;  font-size: 20pt }
</style>
</head><body>
<div class=mybox>Iamque fretum Minyae Pagasaea puppe
secabant.</div>
</body></html>
```



Inline and block boxes

```
<style type="text/css">
.inlinebox { display: Inline; padding: 5mm }
.blockbox { display: block; border: 6pt solid red }
</style></head><body>
    <div class=blockbox>
        <div class=inlinebox>Text in the first left block</div>
        <div class=inlinebox>Text in the first right block</div>
    </div>
    <div class=blockbox>
        <div class=inlinebox>Text in the second left block</div>
        <div class=inlinebox>Text in the second right block</div>
    </div>
```

Text in the first left block Text in the first right block

Text in the second left block Text in the second right block

Text in the first left block Text in the first right block

Text in the second left block Text in the second right block

The display property

block	A block box (laid out vertically).
inline	One or more inline boxes (laid out horizontally). Split into several boxes if all does not fit into the line.
inline-block	a block box, which itself is flowed as a single inline box, similar to a replaced element. The inside of an inline-block is formatted as a block box, and the element itself is formatted as a replaced element on the line.
list-item	• a list of • items
run-in	Either block or in-line depending on context, example: TO BE OR NOT TO BE, that is the question.
table	As HTML <TABLE>
table-row	As HTML <TR>
table-cell	As HTML <TD>

Useful to achieve side-by-side effects, does not work with Internet Explorer.

Static and relative positioning

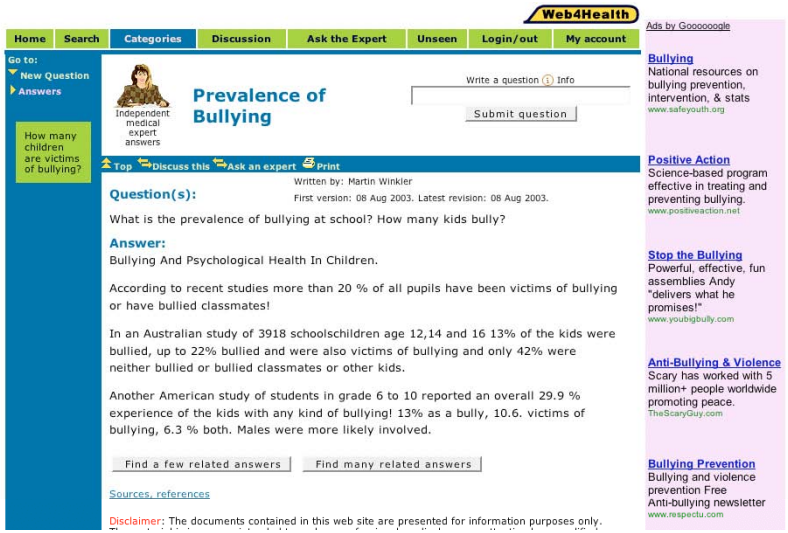
position:

static	Normal flow
relative	Normal flow plus offset
absolute	Fixed position in relation to surrounding block (??)
fixed	Fixed position in relation to: • Window (screen) • Page (printing)

Combined with top, bottom, right, left which can have:

- absolute value
- percentage (of containing block)
- auto
- inherit

Side by side effects:



Side by side: Why does absolute positioning not work?

```
<div style="margin:auto; width: 200px">
   <div style="position:absolute; left: 0px; width:50px">
       <p>Text in column one.</p>
   </div>
   <div style="position:absolute; left: 55px; width:50px">
       <p>Text in column two.</p>
   </div>
   <div style="position:absolute; left: 110px; width:50px">
       <p>Text in column three.</p>
   </div>
</div>
<div id="footer">
   <p>Text below the table
</div>
```

Note:

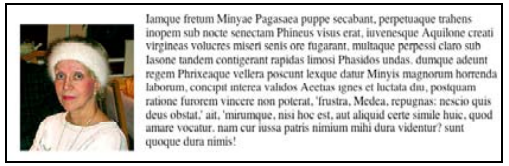
"margin-left" works somewhat better than "left", why?

Side by side: Table cells work with most browsers except Explorer

```
<html><head>
    <style type="text/css">
    .table { display:table; margin: auto; }
    .row { display:table-row; }
    .cell { display:table-cell; width:50px; padding:5px; }
    #footer { text-align:center; }
    </style>
</head><body>
<div class="table">
    <div class="row">
        <div class="cell"><p>Text in column one.</p></div>
        <div class="cell"><p>Text in column two.</p></div>
        <div class="cell"><p>Text in column three.</p></div>
    </div>
</div>
<div id="footer">
    <p>Text below the table
</div>
</body></html>
```


Float

```
<STYLE type="text/css">
IMG { float: left }
BODY, P, IMG { margin: 1em }
</STYLE></HEAD><BODY>
<P>
<IMG src="gunborg-palme-154px.jpg" alt="A floating picture">
Iamque fretum Minyae Pagasaea puppe secabant, perpetuaque
trahens inopem sub nocte senectam Phineus visus erat,
iuvenesque Aquilone creati virgineas volucres miseri senis ore
fugarant, multaque perpessi ... ...
```

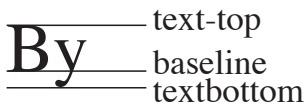


Layering

```
<HTML><HEAD><STYLE type="text/css">
.stackone {
position: absolute; left: 10px; top: 10px; width: 120px; height: 128px }
.stacktwo {
position: absolute; left: 140px; top: 10px; width: 120px; height: 128px }
</STYLE></HEAD><BODY>
<IMG src="eatfem.gif" alt="Eating" class=stackone style="z-index: 2">
<DIV class=stackone style="z-index: 1">
eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat
eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat </DIV>
<IMG src="eatfem.gif" alt="Eating" class=stacktwo style="z-index: 1">
<DIV class=stacktwo style="z-index: 2">
eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat
eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat eat</DIV>
```

vertical-align

```
<IMG src="EATFEM.GIF" style="vertical-align:middle">
Middle: <IMG src="flag.gif" style="vertical-align: middle">
baseline: <IMG src="flag.gif" style="vertical-align: baseline">
text-top: <IMG src="flag.gif" style="vertical-align: text-top">
text-bottom: <IMG src="flag.gif" style="vertical-align: text-bottom">
bottom: <IMG src="flag.gif" style="vertical-align: bottom">
top: <IMG src="flag.gif" style="vertical-align: top">
```



text-indent

```
<style type="text/css">
body { width: 50ex }
p { line-height: 130%; text-indent: 2em;
    padding: 0px; margin: 0px }
</style></head><body>
<p>Iamque fretum Minyae Pagasaea puppe secabant ....</p>
<p>Iuvenesque Aquilone creati virgineas volucres ....</p>
<p>Multaque perpessi claro sub Iasone tandem ....</p>
<p>Dumque adeunt regem Phrixeaque vellera poscunt ....</p>
```

Iamque fretum Minyae Pagasaea puppe secabant, perpetuaque trahens inopem sub nocte senectam Phineus visus erat.

Iuvenesque Aquilone creati virgineas volucres miseri senis ore fugarant.

Multaque perpessi claro sub Iasone tandem contigerant rapidas limosi Phasidos undas.

Dumque adeunt regem Phrixeaque vellera poscunt lexque datur Minyis magnorum horrenda laborum.

Border styles

```
<style type="text/css">
span { padding: 4px;     margin: 4px }
body { margin: 20px; font-family: sans-serif; font-size: 24px }
</style></head><body>
<span style="border:solid 10px">solid</span>
<span style="border:dotted 10px">dotted</span>
<span style="border:dashed 5px">dashed</span>
<span style="border:double 10px">double</span>
<span style="border:groove 10px">groove</span>
<span style="border:ridge 10px">ridge</span>
```

Background

```
background-image:
   url("http://web4health.info/images/wood2.gif");
   background:repeat: no-repeat;
   background-position: center top;
   background-color: #FFCC00;
}
```

Short format

```
background:
   url("http://web4health.info/images/wood2.gif")
   no-repeat center top #FFCC00;
}
```


Custom bullets using CSS backgrounds

```
<style type="text/css">
 <!--
 ul {
    list-style-type: none;
    padding-left: 0;
    margin-left: 0;
 }
 li {
    background: url(
 markflag-2.gif")
 left center no-repeat;
    padding-left: 18px;
    margin-bottom: 10px;
 }
 -->
 </style>
 </head><body>
 <ul>
    <li>First list item
    <li>Second list item
    <li>Third list item
 </ul>
```

- First list item
- Second list item
- Third list item

Marking visited and unvisited links with CSS backgrounds

```
 <style type="text/css"> <!--
 a {
    text-decoration: none;
    color:#333333;
    padding-right:13px;
 }
 a:link {
    background:
    url("redflag.gif") right center no-repeat
 }
 a:visited {
    background:
    url("check-mark.gif")
    right center no-repeat
 }
 --> </style>

 <A HREF="unvisited-link.html">Unvisited
 link</a><br>
 <A HREF="visited-link.html">Visited link</a>
```

Unvisited link
Visited link✓

*:96 Overheads

Part 7a: Hypertext Transfer Protocol (HTTP)

More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 04-06-15 17.35

Hypertext Transfer Protocol (HTTP)

RFC 1945 Hypertext Transfer Protocol -- HTTP/1.0, May 1996 (Informational).

RFC 2068 Hypertext Transfer Protocol -- HTTP/1.1, January 1997 (Proposed standard).

RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1, June 1999 (Draft standard)

Basic HTTP 1.0 steps:

(1) Client establishes a TCP connection to the server

(2) Client gets the response from the server

(3) Server closes the connection

Home page

Definition 1: The page at the root of a web server, i.e. the page with the address `HTTP://www.foo.bar` and nothing more!

Definition 2: The base page for a set of pages in a common area, or belonging to a certain individual.

Some people claim that the correct term for the latter should be Welcome page or Index page.

A very simple HTTP 1.0 connection:

(Commands are text-based, ends with CRLF)

C:	<connects to www.dsv.su.se at port 80>
:	
C:	GET / HTTP/1.0
:	
S:	<HTML>
	S: <HEAD>
	... ...
	S: </HTML>
S:	<closes the connection to the client>

An example of a complete HTTP 1.0 connection

C:	<connects to www.dsv.su.se at port 80>
C:	GET /~jpalme/test/small.html HTTP/1.0
S:	HTTP/1.1 200 OK Date: Mon, 13 Apr 1998 11:13:46 GMT Server: Apache/1.2.4 Last-Modified: Mon, 13 Apr 1998 11:11:31 GMT ETag: "437e5-98-3531f2e3" Content-Length: 152 Accept-Ranges: bytes Connection: close Content-Type: text/html <HTML> <HEAD> <TITLE>A very small file</TITLE> </HEAD> <BODY BGCOLOR="#FFFFCC"> <H1>A very small file</H1> <P>Which ends here! </BODY> </HTML>
S:	<closes the connection to the client>

An example of an actual HTTP request

```
GET /Icons/headhome.gif HTTP/1.1
Host: w4g.dsv.su.se:9800
Accept: image/gif, image/x-xbitmap, image/jpeg,
         image/pjpeg,image/xbm, image/x-jg, */*
Accept-Language: en
Connection: Keep-Alive
Referer: http://w4g.dsv.su.se:9800/
User-Agent: Mozilla/4.0 (compatible; MSIE 4.0; Mac_PowerPC)
UA-OS: MacOS
UA-CPU: PPC
Cookie: currentLocation=2.1; cursor=%401dce3924; userName=
Extension: Security/Remote-Passphrase
```


Explorer, Windows 95 client

```
GET /Icons/headhome.gif HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
          image/x-jg, */*
Referer: http://w4g:9800/
Accept-Language: sv, en
UA-pixels: 1024x768
UA-color: color16
UA-OS: Windows 95
UA-CPU: x86
User-Agent: Mozilla/2.0 (compatible; MSIE 3.0; Windows 95)
Host: w4g:9800
Connection: Keep-Alive
Cookie: currentLocation=2.1; cursor=%401dcda7f3; userName=
```

HTTP message types:

requests and *responses*

HTTP Request format

request-line, format: <request> <request-URI> <HTTP-version>
<headers (can be omitted)>
<blank line>
<body (only needed for a POST request)>

Example:

```
GET /~jpalme/test/small.html HTTP/1.0
```

<- Request comman
<- A blank line

Why is not a full URL needed? Because the connection is already established, which means that host and port has been selected before the GET request is sent.

Request types

In HTTP 1.0:

GET	returns information for the request-URI.	*GET and HEAD should should not modify info,*
HEAD	same as GET, but only returns header.	
POST	sending in new *subordinate* data, like data base updates, annotations, etc.	*but POST may modify info on the server*

In HTTP 1.1:

OPTIONS	queries which options are available for the resource indicated by the request-URI. If request-URI is "*" the server in general is queried. This HTTP feature is not fully defined yet.
PUT	store what is being sent *under the request-URI*. Compare with POST!
DELETE	delete the resource at the request-URI.
TRACE	Send back the request message all the way from the final recipient to the sender, in format `Content-Type:message/http.`
CONNECT	Reserved for future usage by SSL (Secure Sockets Layer) tunnelling (=Proxy should just forward data without being able to understand anything else than host and port).

HTTP Response format

for machine interpretation — *for human user, but need not be displayed by client*

status-line, format: HTTP-version> <response-code> <response-phrase>
<headers (can be omitted)>
<blank line>
Example: <body>

```
HTTP/1.1 200 OK
Date: Mon, 13 Apr 1998 11:13:46 GMT
Content-Type: text/html

<HTML><HEAD>
   <TITLE>A very small file</TITLE>
</HEAD><BODY BGCOLOR="#FFFFCC">
   <H1>A very small file</H1>
   <P>Which ends here!
</BODY></HTML>
```


Status line contains a 3-digit number, the response code

Important response codes:

100 Continue (server wants client to send more info, like body)

200 OK, request suceeded

202 Accepted but processing not ready

300 Multiple choices, *let user choose*

301 Has new permanent URL

302 Has temporarily a new URI

304 Has not been modified (after a conditional GET)

402 Payment required

400 Bad request

401 Unauthorized, include WWW-Authenticate to start authentication

403 Forbidden

404 Not found

405 Method not allowed

500 Internal server error

501 Not implemented

503 Service unavailable

505 HTTP version not supported

Example of response line with a response code:

```
HTTP/1.1 200 OK
```

Preventing Caching

http://developer.apple.com/internet/safati_faq.html

How do I prevent my pages and cookies from being cached?

```
header:("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
                // Date in the past
header("Last-Modified: " . gmdate("D,d M Y H:i:s") . "GMT"):
                // Always modified
header("Cache-Control: no-store, nocache, must-revalidate,
                max-age=0"); // HTTP/1.1
header("Cache-Control: no-store, no-cache, must-revalidate,
                max-age=0"); // HTTP/1.1
header("Pragma: no-cache"): // HTTP/1.0
```


HTTP Cache Control

Command	Used in	Means
Cache-Control: no-cache=field-name[1]	Request	Do not use cached copy
Cache-Control: no-store	Request, Response	Do not store on discs for security reasons (discs might be backed-up)
Cache-Control: max-age=	Request	Client can accept a response which is not older than this time
Cache-Control: min-fresh=	Request	Client wants response which is valid a certain minimum time
Cache-Control: max-stale=	Request	Client can accept stale data
Cache-Control: only-if-cached	Request	Return only cached data, not data from the original server

[1] field-name restricts this command to only certain fields in the HTTP header.

Command	Used in	Means
Cache-Control: public=field-name	Response	Can be cached and later used for someone else
Cache-Control: private=field-name	Response	Can only be cached for use by this particular user
Cache-Control: no-cache	Response	Do not cache anywhere
Cache-Control: no-transform	Response	Do not cache in media-converted format
Cache-Control: must-revalidate	Response	Always revalidate stale cached data
Cache-Control: proxy-revalidate	Response	Same as must-revalidate, but not for user agent caches
Expires: Absolute date	Response	Refresh cache if cached copy is older than this date
Expires: 0	Response	Same as "Expires immediately".
Cache-Control: max-age=	Response	Similar to "Expires"

HTTP header fields (not complete)

Header name	Req	Resp	Body	Explanation
Encoding of content (not the same as MIME Content-Transfer-Encoding), like compression (gzip, compress, deflate), identity=no compression, encryption:				
Content-Encoding			x	An encoding of the resource itself, for example compression. Not MIME encoding.
Transfer-Enoding			x	An encoding added for this transmission only, a property of the message transmission, not the document as stored.
Transfer-Encoding: chunked			x	Chunked (with a separate size indicator for each chunk, in order to be able to send dynamically produced content)
Accept-Encoding	x			Restricts the Content-Encodings to be used

Header name	Req	Resp	Body	Explanation
Natural language headers				
Content-Language			x	Natural language (en=English, en-US=American English, en-cockney, sv = Swedish, sw=Swahili, etc.). Same header in e-mail and HTTP.
Accept-Language	x			Example: `da, en-gb;q=0.8, en;q=0.7` means: "I prefer Danish, but will accept British English and other types of English."

Header name	Req	Resp	Body	Explanation
Range handling (transferring only part of a resource)				
Accept-Ranges: [bytes/none]		x		Accepts sending of only part of a resource.
Range: bytes NN-NN	x			Indicates that only part of a resource is requested.
If-Range: date	x			Send me the whole if modified since a certain date, otherwise send only the parts requested in the Range header

Content-Range			x	Indicates that only part of the whole resource is transmitted. Examples:
				• The first 500 bytes:
				`bytes 0-499/1234`
				• The second 500 bytes:
				`bytes 500-999/1234`
				• All except for the first 500 bytes:
				`bytes 500-1233/1234`

Header name	Req	Resp	Body	Explanation
Conditional HTTP operations and cache control				
If-Modified-Since: Date	x			Download only if client's cached copy is older than the copy at the server.
If-Unmodified-Since: Date	x			Perform only if client's cached copy is newer than the copy at the server, mostly useful in PUT operations.
ETag			x	An entity-flag is a unique identification of one version of a resource. Strong entity-tag = octet-by-octet identical, weak entity-tag = semantically equivalent Used for Cache validation in cases where dated is not suitable.
If-None-Match: * | 1#entity-tag	x			Send if none of the entities which the client already has are current.
If-Range: Date	x			Send the whole of client's cached copy is older, otherwise send only a range.

Continued...

Header name	Req	Resp	Body	Explanation
If-Match: * | 1#entity-tag	x			Do not perform operation if server does not have entity with one of the given tags (* = any entity). Mostly used in PUT operations to avoid updating already modified document.
Vary		x		Which HTTP request header fields have been used by the server to select a version of a document which exists in multiple versions.
Via		x		Trace list of proxies passed by the object
Warning		x		Warning that caching may have modified the resource
Cache-Control		x		Can be *public*, *private*, *no-cache*. *Private* means: Do not cache and supply to anyone else than the current requestor.
Pragma: no-cache		x		Old HTTP 1.0 command equivalent to *Cache-Control: no-cache.*

Header name	Req	Resp	Body	Explanation
Headers with times and dates				
Date	x	x		When message was sent (not creation date). For a message from a proxy server: When it was sent from the originating server.
Age		x		Proxy server says how old (and possibly stale) its copy is
Expires			x	Date after which entity is stale. The special value "0" means *always expired immediately*.
What is the difference between *Cache-Control: no-cache* and *Expires: 0*?				
Last-Modified			x	Last-modification date of resource sent.
Retry-After		x		Server is down, will be up again at date

Header name	Req	Resp	Body	Explanation
Information about the server				
Location		x		With 2xx response codes: Returned and preferred URI. With 301 (permament) and 302 (temporary) response code instructions for automatic redirection of user to a different URI. 301 may even cause updating of out-of-date links automatically.
Server		x		Server software (product name), e.g. `Server: CERN/3.0 libwww/2.17`

Header name	Req	Resp	Body	Explanation
Content negotiation				
Server driven: algorithm in server chooses which content version to send				
Accept	x			Indicates which media types are acceptable. Example: Accept: audio/*; q=0.2, audio/basic means: I prefer audio/basic but can accept other audio types if audio/basic is not available.
Accept-Language	x			
Accept-Encoding	x			

Header name	Req	Resp	Body	Explanation
Agent driven: algorithm in client chooses which content to send				
Alternates		x		Server tells client what it can choose, not yet fully specified in HTTP 1.1
Security control				
WWW-Authenticate		x		Server asks client to authenticate, can contain authentication parameters.
Proxy-Authenticate		x		Proxy server asks client to authenticate, only used one step from proxy to its immediate client.
Content-MD5			x	Digest of body for authentication purposes.
Authorization	x			Client tells server its credentials.

Header name	Req	Resp	Body	Explanation
Header name	**Req**	**Resp**	**Body**	**Explanation**
Allow			x	Servers tells client which methods it supports for the request-URI, e.g. Allow: GET, HEAD.
Content-Length			x	Size of body in octets. For a HEAD request, returns size of body not sent. Content-Length is mandatory in HTTP 1.0, HTTP 1.1 allows several methods, including that the server closes the connection at the end of the resource transmission, and chunked transmission.
Content-Type			x	MIME content-type of body content.
From	x			E-mail address of requestor.
Host	x			Host and port number from the URL given by the original user.
MIME-version	x	x		Same as in e-mail, always “1.0”.

Header name	Req	Resp	Body	Explanation
Expect	x			Instruction from client of special server behaviour. Most common case is: Expect: 100-continue: Client says will not send body until server asks for it. Server replies with 100-continue to indicate that now the client can send the body.
Referer	x			Client tells server where the URI was found. Can be used to check how people link to your server.
User-Agent	x			Client sfotware, e.g. `Mozilla/1.1N.`
Connection	x	x		Control of persistent connections, etc.

Persistent connections

In HTTP 1.0, the server always closed the connection after having sent the requested resource.

In HTTP 1.1, with persistent connections, there are ways for client and server to signal the close of the connection with the HTTP header “Connection: Close”. If there is no such header in HTTP 1.1, they should assume that the connections is not to be closed.

Persistent connections require that the length of a message is not indicated by closing the connection.

Client may pipeline its requests, i.e. send requests before the response has arrived to previous requests.

Servers may have a time-out value after which persistent connections are closed.

Client may close the connection at any time. Clients and servers must be able to recover from asynchronous close events.

Content-Disposition

The Content-Disposition header, defined in RFC 1806, is *not* a part of the HTTP standard, but is widely implemented. It has certain security problems.

According to RFC 1806:

Content-Disposition = [Inline | Attachment] *(; disposition-parameter)

disposition-parameter = filename "=" value | other-parameter

Optimization of delivery in browsers

Progressive rendering

Part of the text or the image is shown while the rest is being downloaded.

Multiple connections

If a user requests a document with several inline image, many browsers will establish multiple connections to download more than one image at the same time. Browsers may also give priority to downloading those images which are in the visible window.

The use of one TCP connection per file can be very inefficient.

*:96 Overheads

Part 8: Directory systems (systems to find addresses of people and organisations): X.500, Whois

Rating: Platform Independent Content Selection (PICS)

More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/~jpalme/internet-course/Int-app-prot-kurs.html`

Last update: 97-07-27 19.30

Directory systems

Information stored in directory systems

Name, e-mail address, phone and fax number, postal address, etc.

Cryptographic certificates: Directory system serve as trusted certificate servers.

Most advanced cryphtographic services: Identification, authorisation, signatures, seals, encryption of information, is based on or uses as a start electronic certifications of the person you are identifying. To be sure that such a certificate is not falsified, crypthographic techniques require a crypthographically secure communcation with the certificate server.

Systems based on manually created directories

X.500: The OSI directory system

LDAP (Lightweight Directory Acess Protocol): Simplified version of X.500 which many manufacturers support or plan to support

Whois: Old, limited Internet protocol

Whois++: New, advanced protocol

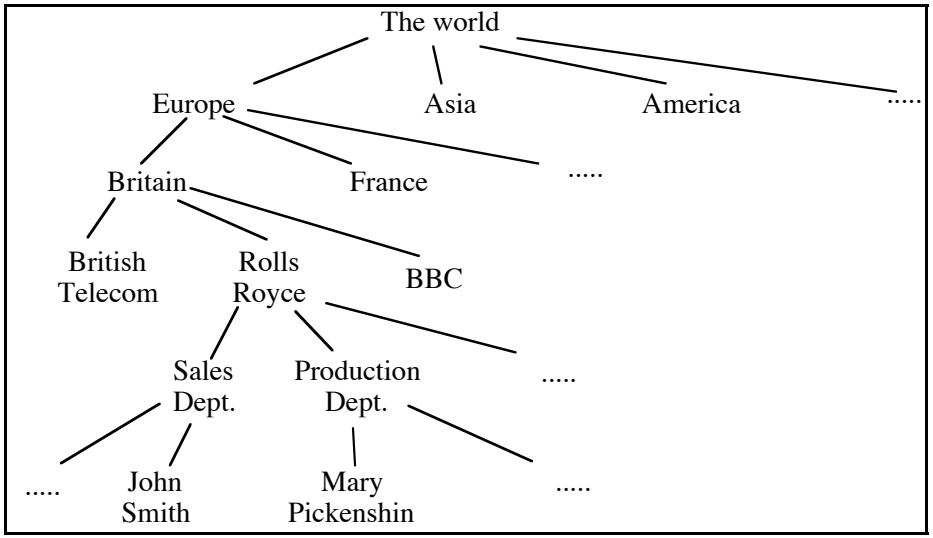
Systems based on automatic collection of information

WhoWhere, etc.: Servers who collect user information from mail, news and web pages.

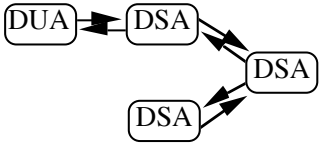
X.500 – the OSI directory system

- Distributed on many servers, like DNS.
- Replication and caching.
- Schema describes data base structure.
- Aliases.

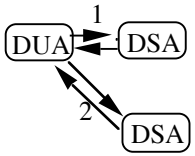
X.500 – hierarchical world view



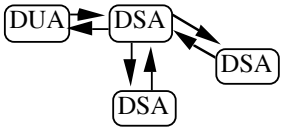
Chaining, referral and multicasting in X.500



This diagram shows *chaining*, where a query is forwarded from DSA to DSA and the reply is returned the same way.

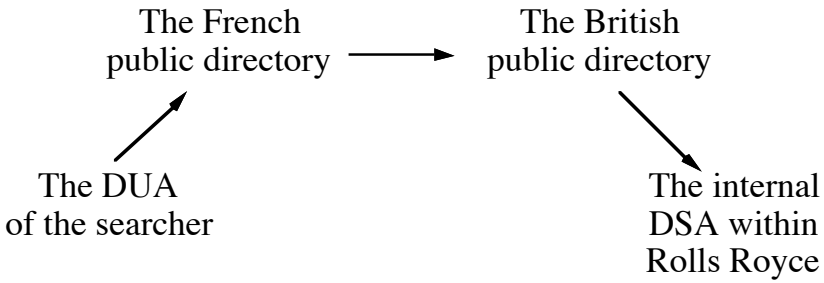


This diagram shows *referral*, where the DUA which sends a query is referred, by the initial DSA, to another DSA which is able to answer the query.



This diagram shows *multicasting*, where the first DSA will simultaneously send the query to several other DSAs and collect the replies.

Chaining in X.500



Sunet X.500 service user interface

8-9

Sunet X.500 service example of response

Move upwards to Data-och systemvetenskap

Found one entry by **exact** match.

Jacob Palme

Name
Jacob Palme
E-Mail
jpalme@dsv.su.se

8-10

Old Internet Whois service

Usually accessed via Gopher and not the Whois protocol itself

Gopher Menu

- Albert Einstein College of Medicine
- Algonquin College of Applied Arts and Technology, Nepean, Ontario, Canada
- American Mathematical Society Combined Membership List
- American University, Washington DC
- Arizona State University
- Auburn University
- Bates College
- Baylor College of Medicine
- Beth Israel Hospital (Harvard Univ.)

8-11

Connection to one Whois server at one university

gopher://ns.bcm.tmc.edu:105/2 CSO Search

A CSO database usually contains a phonebook or directory. Use the search function of your browser to enter search terms.

This is a searchable index. Enter search keywords: john smith

CSO Search Results

name: Smith-Johnson, Gwendolyn M

```
     title: Admin Assistant, L I
department: Medical Illustration
   address: BCM-Cullen Building 303A
  category: Regular Faculty/Staff
     phone: 713-798-4681
       fax: 713-798-6853
     email: gjohnson@bcm.tmc.edu
```

8-12

Main functional difference between Whois++ and X.500:

Searches in tree-structed distributed directory systems like X.500 are difficult to perform efficiently if the user does not specify search conditions which limit the search to certain branches of the tree. If not, a global search has to be made in all servers everywhere.

Because of this, centralized data bases can easier perform efficient directory searches. Distributed and centralized data bases can be combined, if the centralized data bases replicate information whose master copy is in the distributed servers.

The Whois++ developers claim that Whois++ supports such replication of information to central data bases more efficiently than X.500. A subset of the part of a data base which is needed centrally, called a *centroid*, is copied in a controlled way to centralized so-called index servers. *A user search* which is not limited to a certain server, *is helped by the index servers to find the local server which has the directory information* searched for.

Whois++ user interface example

Digger White Pages search

Search for: smith

Show response as: ○ All fields ◉ Expanded list ○ One per line

Choose a server to start search at:
World [services.bunyip.com 63]

Submit Search

If you have any questions, send mail to: digger-info@bunyip.com

---

This Digger server is provided by Comedia Information AB.

Digger is a Registered Trade Mark of Bunyip Information Systems Inc.</H5

Whois++ search result unser interface

Digger search for "smith" , starting search on World level

More records exists, but contact could not be made with MUDDCS.CS.HMC.EDU

More records exists, but contact could not be made with IBS.EIT.COM

More records exists, but contact could not be made with WHOIS.LUT.AC.UK

Hugh Smith
Email: *hugh@nexor.co.uk*
Phone: +44-115-952-0503

Digger is a Registered Trade Mark of Bunyip Information Systems Inc

Whowhere:

http://users.aimnet.com/~dtowner/who.html

Looking for PEOPLE on the Net?

Enter the**Name of Person** you are looking for:
Jacob Palme (REQUIRED)

Enter any information you have about the**Organization** that provides an E-MAIL account for this person. For example , the Organization name and location (city, state or country)
(OPTIONAL)
Search Reset

Looking for ORGANIZATIONS on the Net?

Enter the **Organization Name and Location** information such as city, state or country: Example

Search

WhoWhere search result

Email Search Results: **Over 500 approximate matches**

Name: Jacob Palme **Info:** Search
◉ all matches ○ only exact matches

Highly Relevant Responses

Name: Jacob Palme
E-mail: jpalme@dsv.su.se (click to send email)
Email Provider: University of Stockholm
Last Updated: March '96
Want to know more about Jacob Palme?

Name: Jacob Palme
E-mail: jjalme@mars.dsv.su.se (click to send email)
Email Provider: University of Stockholm
Last Updated: --

Name: Jacob Palme
E-mail: jpalme@heron.dafa.se (click to send email)
Email Provider: Dafa Data Ab
Last Updated: --

Personally registered information in WhoWhere

Jacob Palme
jpalme@dsv.su.se
http://www.dsv.su.se/~jpalme

Primary Email:	jpalme@dsv.su.se
Web Page URL:	http://www.dsv.su.se/~jpalme
Address:	Skeppargatan 73 Stockholm, S-115 30 Sweden
Phone:	+46-8-16 16 67

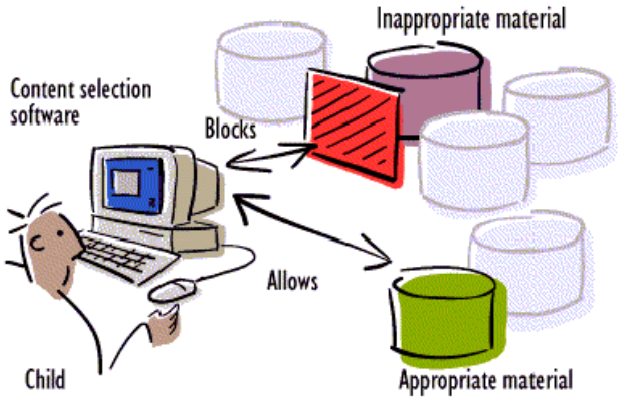
Non-tenured professor of computer science at Stockholm

This Listing was Last Customized in March '96

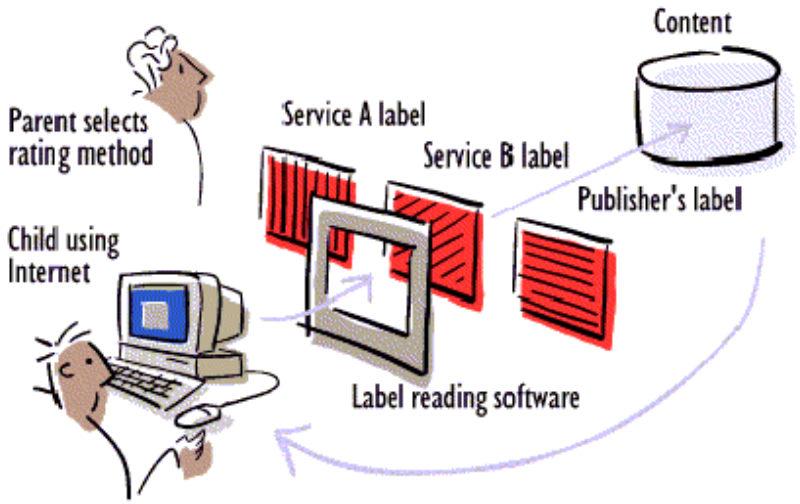
PICS 1 – Platform for Internet Content Selection

From a presentation by Jim Miller and Paul Resnic, found at URL:

`http://www.chg.ru/w3/PICS/951030/JM/talk.htm` and
`http://www.bilkent.edu.tr/pub/WWW/PICS/iacwc.htm`



PICS 2



PICS 3

PICS Technical Mission, excerpt

The technical working group will devise a values-neutral infrastructure for Internet content labeling. The three primary goals are to:

1. enable content providers to voluntarily label the content they create and distribute.
2. enable third-party labeling services to associate additional labels with content created and distributed by others. Services may devise their own labeling systems, and the same content may receive conflicting labels from different services.
3. enable parents and teachers to use the labels to control the information that children under their supervision receive.

PICS 4

A rating *service* is an individual, group, organization or company that produces labels for information. A *rating system* is a way of rating information, consisting of one or more *categories* and a *scale* for each category.

The **Motion Picture Association of America** (MPAA) is a rating service, which uses a well-known (in the United States) rating system for rating movies. Other organizations also provide rating systems or services, such as **SafeSurf** and **SurfWatch** (both PICS founding members) and the **Recreational Software Advisory Council** (a PICS supporting member).

A rating system provides a number of *categories* (or *dimensions*) along which information can be rated.

The MPAA rating system has only one category, the overall rating of the movie. The RSAC rating system has three categories: *nudity/sex*, *violence*, and *language*.

A rating system provides a number of *categories* (or *dimensions*) along which information can be rated.

The MPAA rating system has only one category, the overall rating of the movie. The RSAC rating system has three categories: *nudity/sex*, *violence*, and *language*.

A rating system provides a *scale* for each category.

The MPAA rating system's one category has a scale with values like "G," "PG," and so forth. RSAC's *nudity/sex* category uses a scale with values of "suitable for all ages," "partial nudity," and so on. RSAC's *language* category uses a scale with values of "some profanity," "explicit sexual references" and so on.

PICS Content Labels are values neutral

"The technical working group will define a format for labels, indicating required and optional fields. The format will not specify which words or categories will be used for labeling or the criteria for assigning labels to items."

(excerpt from the Technical Committee Charter)

Required information:

- the **rating service** which created the label,
- the name of a **category** in their **rating system**,
- a **value** on the **scale** for that category.

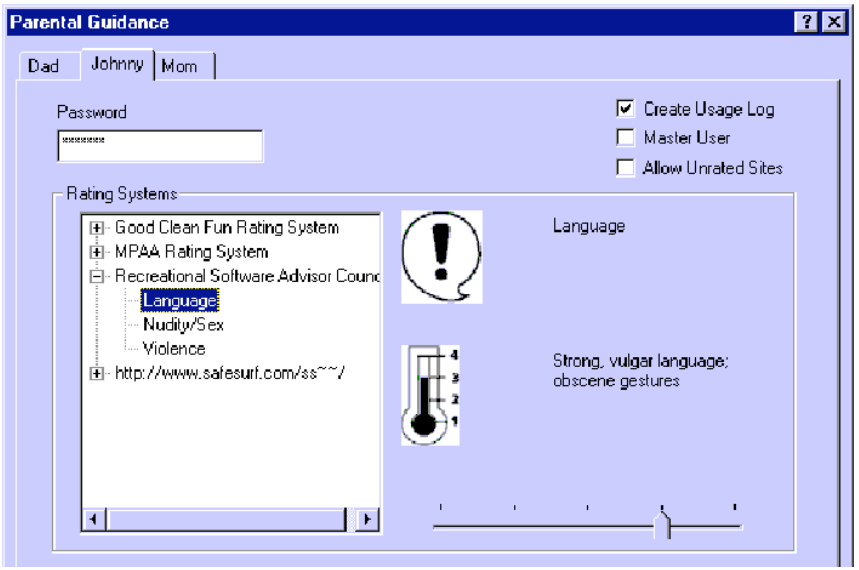
Optional information:

- When the label was assigned to the information
- When the label expires
- Consistency checking information (URL of information, date of information when it was rated, cryptographic checksum of information actually rated)
- Who provided the rating
- Where additional information can be found

Specification of a PICS schema

```
((PICS-version 1.0)
  (rating-system "http://moviescale.org/Ratings/Description/")
  (rating-service "http://moviescale.org/v1.0")
  (icon "icons/moviescale.gif")
  (name "The Movies Rating Service")
  (description
  "A rating service based on the MPAA's movie rating scale")

  (category
   (transmit-as "r")
   (name "Rating")
   (label (name "G") (value 0) (icon "icons/G.gif"))
   (label (name "PG") (value 1) (icon "icons/PG.gif"))
   (label (name "PG-13") (value 2) (icon "icons/PG-13.gif"))
   (label (name "R") (value 3) (icon "icons/R.gif"))
   (label (name "NC-17") (value 4) (icon "icons/NC-17.gif"))))
```


User interface for parents



An example of a PICS label

PICS specifies a standard format for labels. Figure 5 shows a sample. The URL on the first line, which identifies the labeling service, makes it possible to redistribute labels yet still identify their original sources. The label can also include information about itself, such as the date on which it was created, the date it will expire, that the label is associated with a certain resource (in this case, "http://www.gcf.org/stuff.html"), and the label's author. The last line shows the attributes that describe the resource: a "language" value of 3; "sex" 2; and "violence" 0.

```
 (PICS-1.0 "http://www.rsac.org/v1.0/"
 labels
  on "1994.11.05T08:15-0500"
  until "1995.12.31T23:59-0000"
  for "http://www.gcf.org/stuff.html"
  by "John Patrick"
  ratings (l 3 s 2 v 0))
```


An example of a PICS interaction

Client sends to HTTP server www.greatdocs.com to request a document:

```
GET foo.html HTTP/1.1
Accept-Protocol: {PICS-1.0 {params full {services
"http://www.gcf.org/1.0/"}}}
```

Server responds to client with result including PICS label:

```
HTTP/1.1 200 OK
Date: Thursday, 30-Jun-95 17:51:47 GMTMIME-version:
1.0
Last-modified: Thursday, 29-Jun-95 17:51:47 GMT
Protocol: {PICS-1.0 {headers PICS-Label}}
PICS-Label: …label here…
Content-type: text/html
…contents of foo.html…
```

Terminology

VAC	Voluntary Access Control
PICS	Platform for Internet Content Selection
WWWC, W^{3}C	World Wide Web Consortium
Rating service	Service provider providing rating services
Rating system	Schema for a rating service
Content label	The rating of a particular object (document, site, domain)

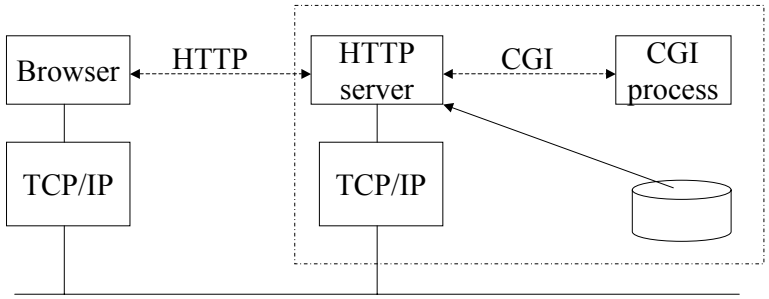
CGI och CGI-programmering

Fredrik Kilander
DSV

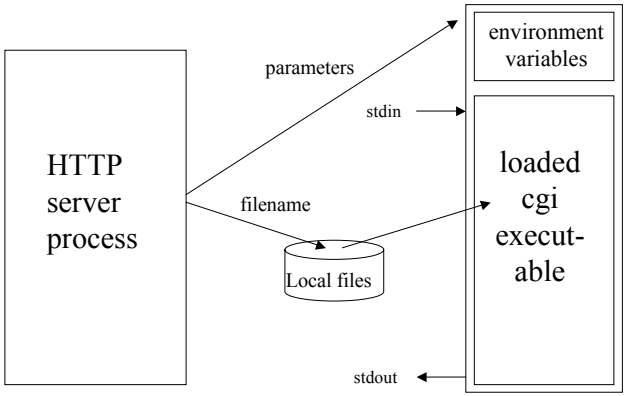
Innehåll

- Common Gateway Interface (CGI)
- Alternativ för dynamiska WWW-sidor
- HTTP-servern
- CGI-processen
- Programmeringsspråk
- Säkerhet
- Applikationsdesign för WWW

Common Gateway Interface



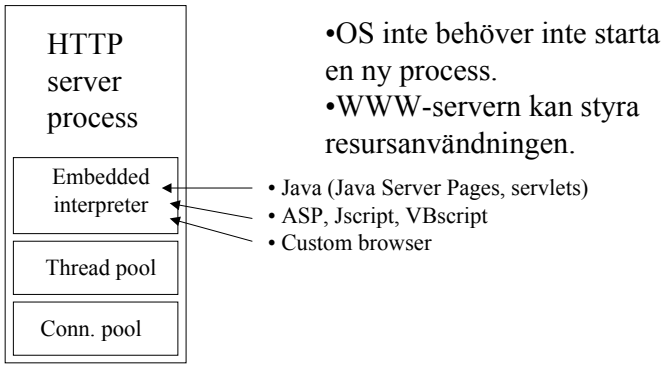
Common Gateway Interface



CGI: en de facto standard

- Informell överenskommelse
- Beskriver server och CGI-program
- Internet draft version 1.1, Juni 1999, (utgången)
- Version 1.2 (inga framsteg sedan 1998).
- http://www.w3.org/CGI
- http://cgi-spec.golux.com

Kompletterande lösningar



•OS inte behöver inte starta en ny process.
•WWW-servern kan styra resursanvändningen.

Dynamiska websidor i WWW-läsaren

- Javascript, JScript, VBscript
- Java (applets)
- ActiveX
- Vendor plug-ins (Flash, Quicktime…)

- CGI är inte beroende av klientimplementationen

HTTP-servern

- WWW-läsaren och servern använder HTTP
- WWW-läsaren anropar med GET eller POST
- GET : hämta URL
- GET : `www.bz.com/db.exe?nm=John+Smith&tel=123+987`
- POST : skicka data till servern och få svar
- POST : `<form action="http://www.bz.com/db.exe" method="POST">`

HTTP-servern

- Skicka URL eller exekvera CGI bestäms av serverns konfiguration.
- Hitta exekverbar fil och skapa en process.
- Initiera miljövariabler (environment).
- Skicka POST data till stdin.
- Starta processen.
- Skicka stdout till klienten.

CGI-programmet

- Läsa miljövariabler.
- Läsa stdin (om POST).
- Avkoda parametrar.
- Formatera och skicka ett svar på stdout.
- Svaret är ett Internet-dokument

CGI-programmet

Läsa miljövariabler

REQUEST_METHOD
QUERY_STRING
CONTENT_LENGTH

Fler miljövariabler

- SERVER_SOFTWARE
- SERVER_NAME
- GATEWAY_INTERFACE
- SERVER_PROTOCOL
- SERVER_PORT
- REQUEST_METHOD
- PATH_INFO
- PATH_TRANSLATED
- SCRIPT_NAME
- QUERY_STRING
- REMOTE_HOST
- REMOTE_ADDR
- AUTH_TYPE
- REMOTE_USER
- REMOTE_IDENT
- CONTENT_TYPE
- CONTENT_LENGTH
- HTTP_ACCEPT
- HTTP_USER_AGENT
- HTTP_*

http://hoohoo.ncsa.uiuc.edu/cgi/env.html

CGI-programmet

- Avkoda parametrarna (URL encoding, RFC 2396)
- GET: `www.bz.com/db.exe?name=%c5ke+%d6rn&age=22`
- POST: `name=%c5ke+%d6rn&age=22`
- Parameterströngen: `name=%c5ke+%d6rn&age=22`

- s ::= namn ’=’ [värde] [’&’ namn ’=’ [värde]] ...

Avkoda CGI-parametrarna

`name=%c5ke+%d6rn&age=22`

1. Dela upp vid ’&’

`name=%c5ke+%d6rn&age=22`

`name=%c5ke+%d6rn`	`age=22`

Avkoda CGI-parametrarna

`name=%c5ke+%d6rn&age=22`

2. Dela upp vid ’=’

`name=%c5ke+%d6rn  age=22`

`name`	`%c5ke+%d6rn`	`age`	`22`

Avkoda CGI-parametrarna

`name=%c5ke+%d6rn&age=22`

3. Byt ’+’ mot ’ ’ (blank)

`name %c5ke+%d6rn age 22`

`name`	`%c5ke %d6rn`	`age`	`22`

Avkoda CGI-parametrarna

```
name=%c5ke+%d6rn&age=22
```

4. Byt ’%xx’ mot tecken

```
name %c5ke %d6rn age 22
```

name	Åke Örn

age	22

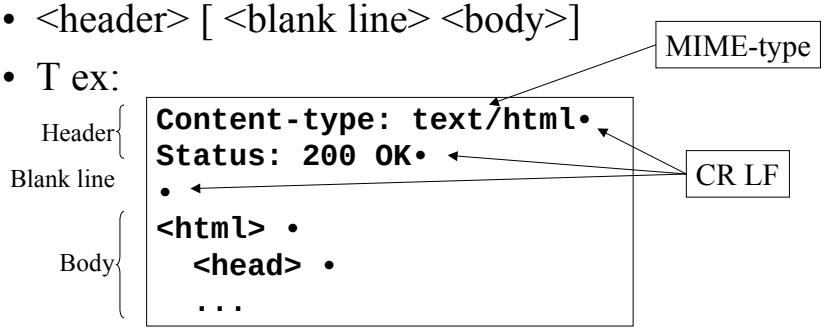
Avkoda CGI-parametrarna

- 1. Dela upp vid ’&’ (par av namn och värden)
- 2. Dela upp vid ’=’ (mellan namn och värde)
- 3. Byt ’+’ mot ’ ’ (blank)
- 4. Byt ’%xx’ mot tecken (hexadecimal kod)

- Färdiga rutiner finns ofta att tillgå

CGI-programmets respons

- Parsed Header Output (HTTP-servern kollar)
- <header> [<blank line> <body>]
- T ex:



Vad ett CGI-script måste göra (v 1.1)

- Avvisa ej understödda metoder* med

```
Status: 405 Method Not Allowed
```

- * GET POST DELETE HEAD PUT OPTIONS TRACE extension-method

Vad ett CGI-script bör göra (v 1.1)

- Vara beredd på att dö närsomhelst (svårt!)
- Hantera PATH_INFO eller svara **`404 Not Found`**
- Verifiera CONTENT_TYPE (indataformat)
- Vara vaksam på ‘//’, ‘.’ och ‘..’ i sökvägar
- Inte generera relativa länkar utan <BASE>
- Sända headerns CGI-fält så snart som möjligt och före HTTP-fält

Programmeringsspråk CGI

- Nästan vilket språk som helst:
- Perl-script (”cgi-script”)
- Shell-script
- C, C++ (lång utvecklingstid)
- Java

Språk inbäddade i HTTP-servern

- Java (Java Server Pages, servlets)
- ASP (Active Server Pages)
- Servermoduler

Java Server Pages

- Källkoden innehåller text till klienten (HTML) och anrop till script-språk (vanligtvis Java).

```
<H1>Welcome to Our Store</H1>
<SMALL>Welcome,
<!-- User name is "New User" for first-time visitors -->
<% out.println(Utils.getUserNameFromCookie(request)); %>
To access your account settings, click
<A HREF="Account-Settings.html">here.</A>
</SMALL>
```

Active Server Pages (ASP)

- Microsoft Internet Information Server IIS
- Körs i HTTP-servern (snabb start)
- Blandar flera språk och syntaxer i samma källkod:
- ASP-script, VBScript, JScript, SSI (servern), VBScript, JScript, JavaScript, HTML, CSS (klienten).

Active Server Pages (ASP)

- + snabb utveckling
- + samma funktionalitet som CGI
- + bra stöd för databaskopplingar (VBscript)
- - kan hänga servern
- - svårt att avlusa
- - ingen modularitet, trasslig syntax
- - klarar inte hög belastning

Servermoduler

- + total kontroll
- + effektivt
- - lång utveckling
- - kan hänga servern
- - kan vara svårt att avlusa
- - dålig portabilitet pga bindning till server

Säkerhet

- Det som CGI-programmet får göra kan också besökarna göra.
- Förhindra ”buffer overruns” (rutinbibliotek)
- Kolla alla indata, inklusive CONTENT_LENGTH.
- Låt aldrig indata exekveras utan inspektion.
- Förbjud allt och släpp in det som är tillåtet, inte tvärtom.
- Logga, och analysera loggarna.

Applikationsdesign för WWW

- WWW är i grunden tillståndslöst.
- Tillstånd behövs i t ex dialoger.
- Cookies, dolda fält

Applikationsdesign för WWW

- Var försiktig med finesser
- Hur stor är användargruppen?
- Vilken utrustning har de?
- Vilka krav ställer applikationen på dem?
- Vad kan man förvänta sig av dem?

Applikationsdesign för WWW

- Enkla användargränssnitt:
- Använd inte mer teknik än som krävs.
- Undvik skärmrullning.
- Använd multipla indikatorer: ledtexter, färger, bilder

Applikationsdesign för WWW

- Grundlig HTML-kodning:
- Följ en (1) standard.
- Stäng alla markörer som får stängas (XHTML).
- Koda speglade indata i formulär.
- Använd analysverktyg.
- Kolla med många www-klienter.

XSL

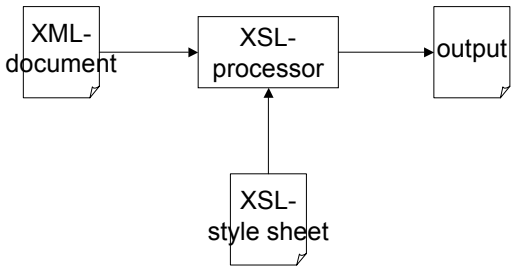
The
Extensible Stylesheet Language
Family

XSL parts

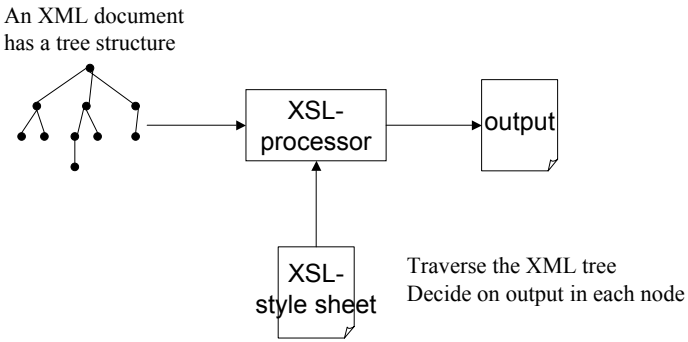
- XSL Transformations (XSLT)
- XML Path Language (XPath)
- XSL Formatting Objects (XSL-FO)

`http:www.w3c.org/Style/XSL`

XSL process view

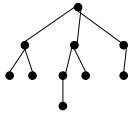


XSL conceptual view



XSL conceptual view

An XML document has a tree structure



XSL-style sheet

The programmer writes a set of templates.

Each template may match zero, one or more nodes.

The body of a template may specify output, or other templates to try in sequence.

The body of a template may also select any part of the tree and generate data from it, or apply templates to it.

The syntax is XML.

XML sample

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<acas:context-state id="global" xmlns:acas="urn:acas:%2f%2fdsv.su.se">

  <acas:contextelement>
    <acas:value datatype="float" unit="celcius">21.2</acas:value>
    <acas:time>2004-04-04 18:57 CET</acas:time>
    <acas:source uri="acas://dsv.su.se">//sensors/k2lab/t2</acas:source>
  </acas:contextelement>

  <acas:contextelement>
    <acas:value datatype="float" unit="celcius">20.7</acas:value>
    <acas:time>2004-04-04 18:58 CET</acas:time>
    <acas:source uri="acas://dsv.su.se">//sensors/k2lab/t1</acas:source>
  </acas:contextelement>

</acas:context-state>
```

XML sample

Namespace

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<acas:context-state id="global" xmlns:acas="urn:acas:%2f%2fdsv.su.se">

  <acas:contextelement>
    <acas:value datatype="float" unit="celcius">21.2</acas:value>
    <acas:time>2004-04-04 18:57 CET</acas:time>
    <acas:source uri="acas://dsv.su.se">//sensors/k2lab/t2</acas:source>
  </acas:contextelement>

  <acas:contextelement>
    <acas:value datatype="float" unit="celcius">20.7</acas:value>
    <acas:time>2004-04-04 18:58 CET</acas:time>
    <acas:source uri="acas://dsv.su.se">//sensors/k2lab/t1</acas:source>
  </acas:contextelement>

</acas:context-state>
```

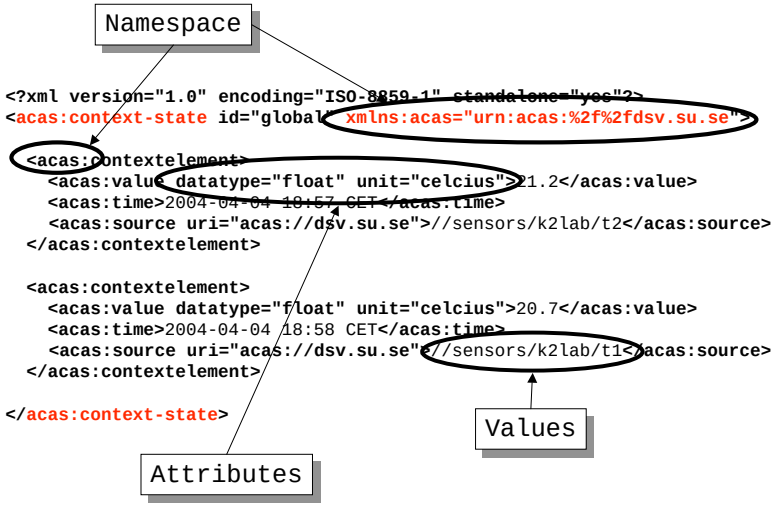
XML sample

Namespace

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<acas:context-state id="global" xmlns:acas="urn:acas:%2f%2fdsv.su.se">

  <acas:contextelement>
    <acas:value datatype="float" unit="celcius">21.2</acas:value>
    <acas:time>2004-04-04 18:57 CET</acas:time>
    <acas:source uri="acas://dsv.su.se">//sensors/k2lab/t2</acas:source>
  </acas:contextelement>

  <acas:contextelement>
    <acas:value datatype="float" unit="celcius">20.7</acas:value>
    <acas:time>2004-04-04 18:58 CET</acas:time>
    <acas:source uri="acas://dsv.su.se">//sensors/k2lab/t1</acas:source>
  </acas:contextelement>

</acas:context-state>
```

Attributes

XML sample



XSLT sample

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- q_one.xsl -->
<!-- 08-mar-2004/FK -->
<xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                exclude-result-prefixes="acas"
                version="1.0">

  <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

  <xsl:template match="/acas:context-state">
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='geo']"/>
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='celcius']"/>
  </xsl:template>

  <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
    <xsl:text>LOCATION OF </xsl:text>
    <xsl:value-of select="../acas:source"/>
    <xsl:text> IS </xsl:text>
    <xsl:value-of select="."/>
    <xsl:text>&#xA;</xsl:text>
  </xsl:template>
```

XSLT sample

The xsl:stylesheet element wraps the whole document

```
  <?xml version="1.0" encoding="iso-8859-1"?>
  <!-- q_one.xsl -->
  <!-- 08-mar-2004/FK -->
  <xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                  exclude-result-prefixes="acas"
                  version="1.0">

    <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

    <xsl:template match="/acas:context-state">
      <xsl:apply-templates
           select="acas:contextelement/acas:value[@unit='geo']"/>
      <xsl:apply-templates
           select="acas:contextelement/acas:value[@unit='celcius']"/>
    </xsl:template>

    <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
      <xsl:text>LOCATION OF </xsl:text>
      <xsl:value-of select="../acas:source"/>
      <xsl:text> IS </xsl:text>
      <xsl:value-of select="."/>
      <xsl:text>&#xA;</xsl:text>
    </xsl:template>
```

XSLT sample

Declaration of namespace in input XML

```
  <?xml version="1.0" encoding="iso-8859-1"?>
  <!-- q_one.xsl -->
  <!-- 08-mar-2004/FK -->
  <xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                  exclude-result-prefixes="acas"
                  version="1.0">

    <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

    <xsl:template match="/acas:context-state">
      <xsl:apply-templates
           select="acas:contextelement/acas:value[@unit='geo']"/>
      <xsl:apply-templates
           select="acas:contextelement/acas:value[@unit='celcius']"/>
    </xsl:template>

    <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
      <xsl:text>LOCATION OF </xsl:text>
      <xsl:value-of select="../acas:source"/>
      <xsl:text> IS </xsl:text>
      <xsl:value-of select="."/>
      <xsl:text>&#xA;</xsl:text>
    </xsl:template>
```

XSLT sample

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- q_one.xsl -->
<!-- 08-mar-2004/FK -->
<xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                exclude-result-prefixes="acas"
                version="1.0">

  <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

  <xsl:template match="/acas:context-state">
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='geo']"/>
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='celcius']"/>
  </xsl:template>

  <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
    <xsl:text>LOCATION OF </xsl:text>
    <xsl:value-of select="../acas:source"/>
    <xsl:text> IS </xsl:text>
    <xsl:value-of select="."/>
    <xsl:text>&#xA;</xsl:text>
  </xsl:template>
```

Type of output

XSLT sample

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- q_one.xsl -->
<!-- 08-mar-2004/FK -->
<xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                exclude-result-prefixes="acas"
                version="1.0">

  <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

  <xsl:template match="/acas:context-state">
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='geo']"/>
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='celcius']"/>
  </xsl:template>

  <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
    <xsl:text>LOCATION OF </xsl:text>
    <xsl:value-of select="../acas:source"/>
    <xsl:text> IS </xsl:text>
    <xsl:value-of select="."/>
    <xsl:text>&#xA;</xsl:text>
  </xsl:template>
```

Template to match root of input tree

XSLT sample

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- q_one.xsl -->
<!-- 08-mar-2004/FK -->
<xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                exclude-result-prefixes="acas"
                version="1.0">

  <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

  <xsl:template match="/acas:context-state">
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='geo']"/>
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='celcius']"/>
  </xsl:template>

  <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
    <xsl:text>LOCATION OF </xsl:text>
    <xsl:value-of select="../acas:source"/>
    <xsl:text> IS </xsl:text>
    <xsl:value-of select="."/>
    <xsl:text>&#xA;</xsl:text>
  </xsl:template>
```

Expressions that select and compose are written in the XPath language.

XSLT sample

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- q_one.xsl -->
<!-- 08-mar-2004/FK -->
<xsl:stylesheet xmlns:acas="urn:acas:%2f%2fdsv.su.se"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                exclude-result-prefixes="acas"
                version="1.0">

  <xsl:output method="text" encoding="iso-8859-1" standalone="yes"/>

  <xsl:template match="/acas:context-state">
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='geo']"/>
    <xsl:apply-templates
         select="acas:contextelement/acas:value[@unit='celcius']"/>
  </xsl:template>

  <xsl:template match="acas:contextelement/acas:value[@unit='geo']">
    <xsl:text>LOCATION OF </xsl:text>
    <xsl:value-of select="../acas:source"/>
    <xsl:text> IS </xsl:text>
    <xsl:value-of select="."/>
    <xsl:text>&#xA;</xsl:text>
  </xsl:template>
```

Output can be constants in the stylesheet or derived from the input.

XSLT element samples

- xsl:template - *defines a template*
- xsl:value-of - *take the value of an XPath expression*
- xsl:text - *text constant*
- xsl:apply-templates - *recurse on a selected set of nodes*

xsl:template

```
<order>
  <id>1234</id>
  <customer>Alice</customer>
  ...
</order>
```

```
<xsl:template match="order">
  ...
</xsl:template>
```

xsl:apply-templates

```
<order>
  <id>1234</id>
  <customer>Alice</customer>
  ...
</order>
```

```
<xsl:template match="order">
  <xsl:apply-templates select="customer"/>
</xsl:template>
```

xsl:value-of

```
<order>
  <id>1234</id>
  <customer>Alice</customer>
  ...
</order>
```

```
<xsl:template match="order">
  <xsl:apply-templates select="customer"/>
</xsl:template>

<xsl:template match="customer">
  <xsl:value-of select="."/>
</xsl:template>
```

xpath

```
<xsl:template match="order">
  <xsl:apply-templates select="customer"/>
</xsl:template>

 <xsl:template match="customer">
   <xsl:value-of select="."/>
 </xsl:template>
```

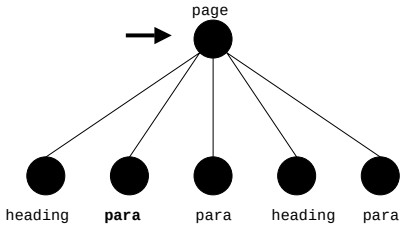
Axis, location paths, expressions

xpath axis

- ancestor
- ancestor-or-self
- attribute
- child
- descendent
- descendent-or-self
- following
- following-sibling
- namespace
- parent
- preceding
- preceding-sibling
- self

```
Axis are used in location paths.

Paths can be absolute or relative.


Axis describe subsets of nodes.

Axis + expression = node set
```

xpath expression




```
<page>
  <heading>...</heading>
  <para>...</para>
  <para>...</para>
  <heading>...</heading>
  <para>...</para>
</page>
```

```
... select="child::para[position()=1]" ...
```

xpath core function library

- *number* last()
- *number* position()
- *number* count(*node-set*)
- *string* local-name(*node-set*?)
- *string* namespace-uri(*node-set*?)
- *string* name(*node-set*?)
- *string* string(*object*?)
- *string* concat(*string, string, string**)
- *boolean* starts-with(*string, string*)
- *boolean* contains(*string, string*)
- *string* substring-before(*string, string*)
- ...

XSL in Java

- javax.xml.parsers.*
- javax.xml.transform.*
- org.w3c.dom.*
- org.xml.sax.*

```
All in Java SDK 1.4
```

XML börjar på sid 26

Example of XML using these DTDs

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE collection SYSTEM
"http://www.dsv.su.se/~jpalme/internet-course/xml/collection.dtd">
<collection
  owner="Kungliga Biblioteket"
>
<book
  title="False Pretences"
  author="Margaret Yorke"
  format="hard-back"
/>
<book
  title="Act of Violence"
  author="Margaret Yorke"
  format="paper-back"
/>
</collection>
```


IDs in XML

Unique names can be used to refer between different places in a document.

XML example:

```
<author ref="myorke">Margaret Yorke</author>
...
<book author="myorke">False Pretences</book>
```

Based on the DTD:

```
<!ELEMENT author (#PCDATA)>
<!ATTLIST author
  ref ID #REQUIRED>
<!ELEMENT book (#PCDATA)>
<!ATTLIST book
  author IDREF #IMPLIED>
```

Attribute types:

ID = Name of this object

IDREF = One single ID reference

IDREFS = List of names separated by white space

NMTOKEN, NMTOKENS = Single words or lists of words separated by white space

Name Spaces

Part of the war DTD:

```
<!ELEMENT war:desert
(deserter*)>
<!ELEMENT war:deserter
(#PCDATA)>
```

Part of the geography DTD:

```
<!ELEMENT geography:desert
(#PCDATA)>
```

Use of these two DTDs in a new DTD desertaions-in-deserts:

```
<!ENTITY % war:desert SYSTEM
"war.dtd">
%war;
<!ENTITY % geography:desert
SYSTEM "geography.dtd">
%geography;
<!ELEMENT desertations-in-
deserts (war:desert,
geography:desert)>
<!ATTLIST desertaions-in-deserts
xmlns:war CDATA #IMPLIED
xmlns:geography CDATA #IMPLIED>
```

XML data:

```
<?xml version="1.0" ?>
<!DOCTYPE desertations-in-deserts
SYSTEM "desertations-in-
deserts.dtd">
<desertations-in-deserts
xmlns:war="http://dsv.su.se/jpalme
/a-book/xml/war.dtd"
xmlns:geography="http://dsv.su.se/
jpalme/a-book/xml/geography.dtd">
  <war:desert>
    <deserter>
    John Smith</deserter>
  </war:desert>
  <geography:desert>
    Sahara</geography:desert>
</desertations-in-deserts>
```

The **xmlns:war="http://dsv.su.se/jpalme/a-book/xml/war.dtd"** and **xmlns:geography="http://dsv.su.se/jpalme/a-book/xml/geography.dtd"** attributes need not refer to any real file, but should contain a unique URL for this name space.

Putting binary data into XML encodings

All textual encodings have a common problem in that they will not allow binary data, like, for example, a picture in GIF format. There are three ways of handling this problem in XML:

① Encode the binary data, using, for example, the BASE64 method.

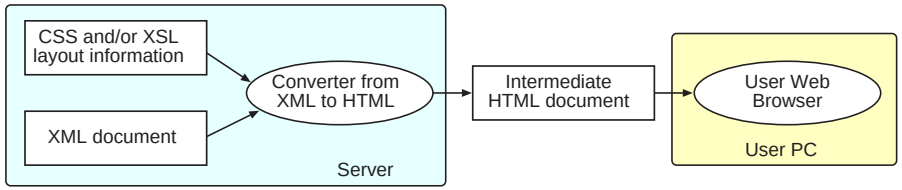
② Put the binary data in a separate file, like GIF pictures in HTML:
<IMG SRC="image.gif">

③ Use method ②, but combine it with the MHTML method to concatenate all the files into a single compound file.

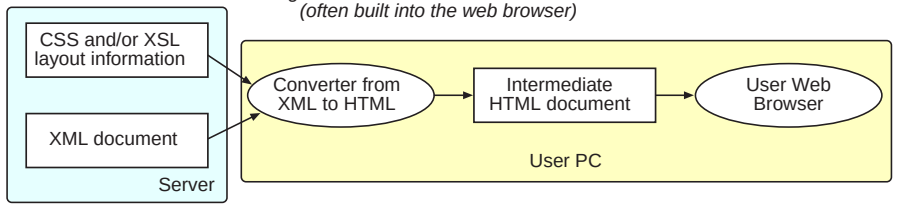
Putting formatting information into XML pages

CSS = Cascading Style Sheets and XSLT = Extensible Style Language Transformations

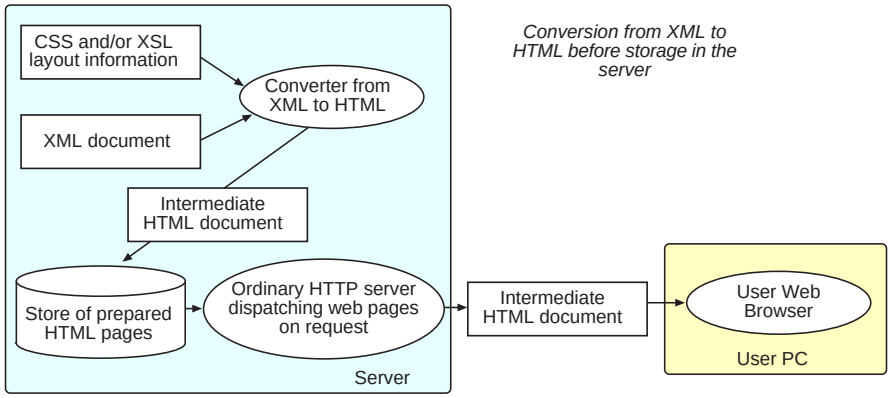
Conversion from XML to HTML in the server, before transmission to the PC



Sending XML to the PC and conversion in the PC (often built into the web browser)



Putting formatting information into XML pages



Conversion from XML to HTML before storage in the server

File ticket.css:

```
TITLE     {    position: absolute; width: 121px; height: 31px; top:25px; left: 86px;
          font-family: Verdana, sans-serif; font-size: 24pt; font-weight: bold}
CLASS     {    position: absolute; width: 106px; height: 15px; top: 115px; left: 13px;
          font-family: Verdana, sans-serif; font-size: 12pt; font-weight: bold }
FROM {    position: absolute; width: 150px; height: 15px; top: 70px; left: 12px;
          font-family: Verdana, sans-serif; font-size: 14pt; font-weight: bold }
TO   {    position: absolute; width: 150px; height: 15px; top: 70px; left: 166px;
          font-family: Verdana, sans-serif; font-size: 14pt; font-weight: bold; }
DEPART    {    position: absolute; width: 142px; height: 15px; top: 95px; left: 11px;
          font-family: Verdana, sans-serif; font-size: 10pt }
ARRIVE    {    position: absolute; width: 128px; height: 15px; top: 95px; left: 167px;
          font-family: Verdana, sans-serif; font-size: 10pt }
CABIN     {    position: absolute; width: 138px; height: 18px; top: 115px; left: 167px;
          font-family: Verdana, sans-serif; font-size: 12pt; font-weight: bold }
SEAT {    position: absolute; width: 138px; height: 18px; top: 115px; left: 247px;
          font-family: Verdana, sans-serif; font-size: 12pt; font-weight: bold }
```

File ticket.xml:

```
<?xml version="1.0" ?>
<!DOCTYPE TICKET SYSTEM "ticket.dtd">
<?XML:stylesheet type="text/css"
href="ticket.css" ?>
<TICKET><TITLE>TICKET</TITLE>
<CLASS>2 Class</CLASS>
<FROM>Oslo</FROM>
<TO>Stockholm</TO>
<DEPART>Mon 13 Jan 12:13</DEPART>
<ARRIVE>Mon 13 Jan 18:45</ARRIVE>
<CABIN>Cabin 3</CABIN>
<SEAT>Seat 55</SEAT></TICKET>
```

Visual rendering:

TICKET

Oslo Stockholm

Mon 13 Jan 12:13 Mon 13 Jan 18:45

2 Class Cabin 3 Seat 5

XML validation

When you are developing specifications using DTD and XML, it is essential to be able to check your specifications for correctness. There is software available to do this. I have been using the validator on the net at http://www.stg.brown.edu/service/xmlvalid/ to validate the examples given in this course.

XHMTL

XHTML is a variant of HTML which is at the same time also correct XML. The main differences from ordinary HTML are:

- All tags must be lower case, e.g. **<a href>** and not **<A HREF=>**
- All tags must be ended, e.g. First paragraph
second line.
- No syntax errors allowed, e.g. not ** ** Strong text ** **

ABNF specification:	ASN.1 specification:	DTD specification:

`Family = "Family"`
`CRLF *(Person)`
`"End of Family"`
 	
`Person = "Person" CRLF`
`" Name: " 1*A CRLF`
`" Birthyear: " 4D CRLF`
`" Gender: "`
`("Male"/"Female") CRLF`
`" Status: "`
`("unmarried"/ "married"/`
`"divorced"/ "widow"/`
`"widower" )`
	Family ::= SEQUENCE OF Person
	
	Person ::= SEQUENCE {
	name VisibleString,
	birthyear INTEGER,
	gender Gender,
	status Status }
	
	Gender ::= ENUMERATED {
	male(0), female(1) }
	
	Status ::= ENUMERATED {
	unmarried(0), married(1),
	divorced(2), widow(3), widower(4)
	}
		`<!ELEMENT family`
		`(person+)>`
		`<!ELEMENT person (name,`
		`birthyear)>`
		`<!ELEMENT name (#PCDATA)>`
		`<!ELEMENT birthyear`
		`(#PCDATA)>`
		`<!ATTLIST person`
		`gender ( male | female )`
		`#REQUIRED`
		`status ( unmarried |`
		`married | divorced |`
		`widow | widower )`
		`#REQUIRED`
		`>`

Example of textual encoding:	Example of BER encoding:	Example of XML encoding:

`Family`
`Person`
`Name: John Smith`
`Birthyear: 1958`
`Gender: Male`
`Status: Married`
`Person`
`Name: Eliza Tennyson`
`Birthyear: 1959`
`Gender: Female`
`Status: Married`
`End of Family`
	(Each box represents one octet. Two-character codes are hexadecimal numbers, one character codes are characters)
	30 34
	30 16
	1A 0A J o h n ␣ S m i t h
	02 02 07 A6
	0A 01 00
	0A 01 01
	30 1A
	1A 0E E l i z a ␣ T e n n y s o n
	02 02 07 A7
	0A 01 01
	0A 01 01
		`<?xml version="1.0" ?>`
		`<!DOCTYPE family SYSTEM "family.dtd">`
		`<family>`
		`<person gender="male"`
		`status="married">`
		`<name>John Smith</name>`
		`<birthyear>1958`
		`</birthyear>`
		`</person>`
		`<person gender="female"`
		`status="married">`
		`<name>Eliza`
		`Tennyson</name>`
		`<birthyear>1959`
		`</birthyear>`
		`</person>`
		`</family>`

169 octets (excluding newlines)	54 octets	276 octets (excluding newlines and leading spaces)
18 % efficiency	57 % efficiency	11 % efficiency[1]

The PER (unaligned variant) encoding of the same ASN.1 and the same data would be the following 31 octets:

```
00000010   (number of persons in family)
00001010   (10 characters)
 1001010   J
1 101111   o
11 01000   h
110 1110   n
0100 000
10100 11   S
110110 1   m
1101001    i
 1110100   t
1 101000   h
00 000010  (2 octets)
00 00011110   100110   (1958)
0          (male)
 0 01      (married)

000011 10  (14 characters)
100010 1   E
1101100    l
 1101001   i
1 111010   z
11 00001   a
010 0000
1010 100   T
11001 01   e
110111 0   n
1101110    n
1111001    y
1 110011   s
11 01111   o
110 1110   n
0000 0010     (2 bytes)
0000 01111010 0111    (1959)
1          (female)
001        (married)
```

Comparison of ABNF, ASN.1-BER and DTD-XML

	ABNF	ASN.1	DTD+XML
Level	Low level, almost any text.	High level, strongly typed.	High level, but not as good typing facilities as ASN.1.
Encoded form.	Text.	BER,. PER, etc.	Text.
Readability of metalanguage	OK.	Good.	Acceptable.
Readability of encoded data	Very good.	Very bad unless special reader program is used.	Very good.
Efficiency of data packing.	Usually not so good.	About 50 % with BER, almost 100 % with PER.	Not so good.
Binary data	Must be encoded, for example using BASE64.	Can easily be included as is.	Must be encoded, for example using BASE64, or sent as separate files.
Layout facilities	None, but the high freedom allows specification of rather readable formats.	None.	Can be combined with layout languages.

Other Encoding Languages

ABNF, ASN.1 and XML are not the only encoding languages. Some other existing languages are Corba and XDR (External Data Representation). Corba is more programmer-oriented, and provides a programming API for transmission of data between applications running on different hosts. And some protocols, for example the Domain Naming System (DNS) do not use any encoding language at all, their encodings are specified in the form of English-language text and tables.

More information about XML

The official XML standards specification (rather difficult to read):

`http://www.w3.org/TR/REC-xml`

Norman Walsh's XML tutorial:

`http://www.xml.com/xml/pub/98/10/guide1.html`

Rolf Pfeiffer's XML tutorial:

`http://www.software.ibm.com/developer/education/tutorial-prog/abstract.html`

Doug Tidwell's XML tutorial:

`http://www.software.ibm.com/developer/education/xmlintro/`

Validator of DTD/XML encodings:

`http://www.stg.brown.edu/service/xmlvalid/`

XML books, like for example:

XML Bible, by Eliott Rusty Harold, IDG Books, Foster City, CA, U.S.A., 1999.

*:96 Overheads

Part 7b: Cookies

More about this course about Internet application protocols can be found at URL:

`http://www.dsv.su.se/jpalme/internet-course/Int-app-prot-kurs.html`

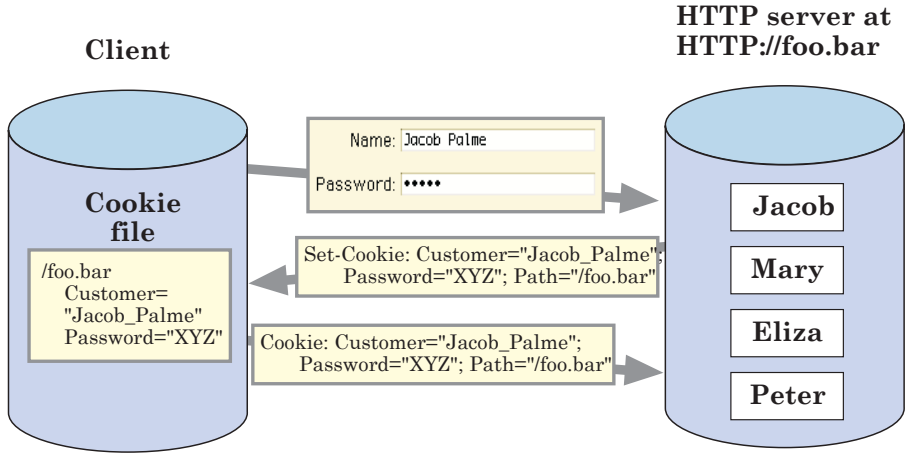
One cookie standard (not entirely accepted by the market):

RFC 2109: HTTP State Management Mechanism

This description follows RFC 2109, even though that may not agree entirely with what is used on the market today.

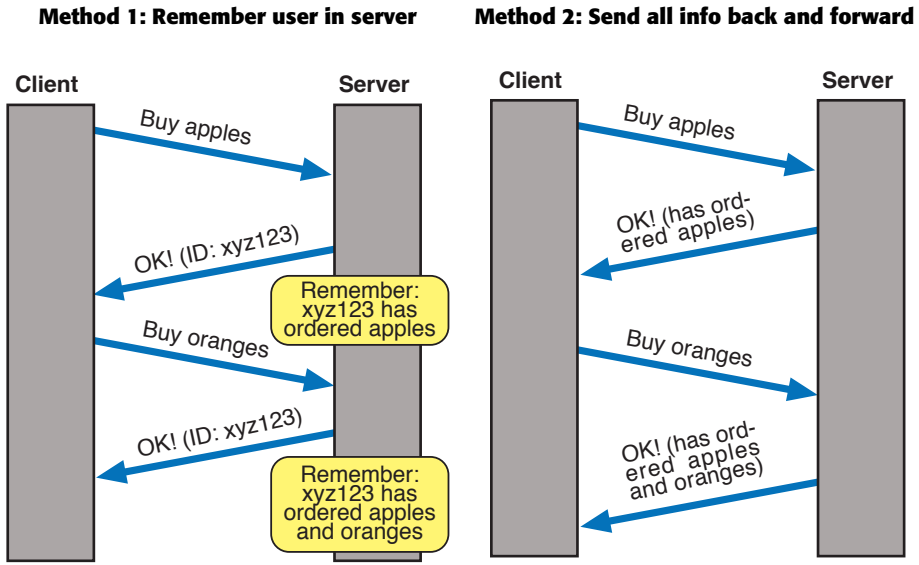
Last update: 05-02-06 17.30

What is a cookie



A cookie is a small piece of data (max 4kB), which the server can store in the client, and which the client sends the next time it connects to the same server to identify itself.

Two ways of remembering what a user did earlier



Example of use of cookies in HTTP transactions

User Server	HTTP command (abbreviated)
→	POST /foo.bar/login HTTP/1.1 [form data with user identification]
←	HTTP/1.1 200 OK Set-Cookie: Customer="JACOB_PALME"; Version="1"; Path="/foo.bar"
→	POST /foo.bar/pickitem HTTP/1.1 Cookie: $Version="1"; Customer="JACOB_PALME"; $Path="/foo.bar" [form data with selection of an item from a shopping basket]
←	HTTP/1.1 200 OK Set-Cookie: Part_Number="Apples-0154"; Version="1"; Path="/foo.bar"
→	POST /foo.bar/shipping HTTP/1.1 Cookie: $Version="1"; Customer="JACOB_PALME"; $Path="/foo.bar"; Part_Number="Rocket_Launcher_0001"; $Path="/foo.bar" [form data]

Cookie management

Creation of cookies

- By client to reside in the client
- By server to reside in the client using the Set-Cookie command
- By server to reside in the server

Usage of cookies

Client sends the cookie when accessing the same server or domain again, using the Cookie header field in the HTTP request.

Set-Cookie header from server to client

Atribute	Description
Name of the cookie, may not start with $	The value of the cookie.
Comment	Explanation for the user, so that the user can decide whether to accept the cookie.
Domain	The domain for which the cookie is valid. Default: the server used. Makes a cookie accessible to several servers in the same domain.
Max-Age	Life time in seconds. Default: When the browser exits. Value 0 can be used to delete an existing cookie.
Path	To which URLs this cookie applies. Restricts which servers can retrieve the cookie.
Secure	Send cookie only using security features.
Version	Version of the standard used.

Privacy issues with cookies

Server can keep track of how often a user accesses the server and what the user does. Can for example be used to select banner advertisment according to the interest of the user. Or can be used to guess at the political opinion of the user, and then used to target political advertising or harassment.

These profiles might be exchanged between servers or sold for profit.

Possible security holes might let the server fetch and modify information it should not have?

Users can set their browsers to reject cookies, or to ask the user before accepting them. But if you reject cookies, you lose a lot of functionality.

There are programs available to help users control and manage their cookies.

XML Validation Form

1. Combine the DTD and XML file into a single file as described in http://dsv.su.se/jpalme/internet-course/combine-dtd-xml.html
2. Click [Browse] to locate this file.
3. Click [Validate] to get it checked.

Local file:
Browse
Suppress warning messages
Relax namespace checks
Validate Clear

1. Put both the DTD and the XML file in the WWW directory on a web server.
2. Click [Validate] to get it checked.

URI:
Suppress warning messages
Relax namespace checks
Validate Clear

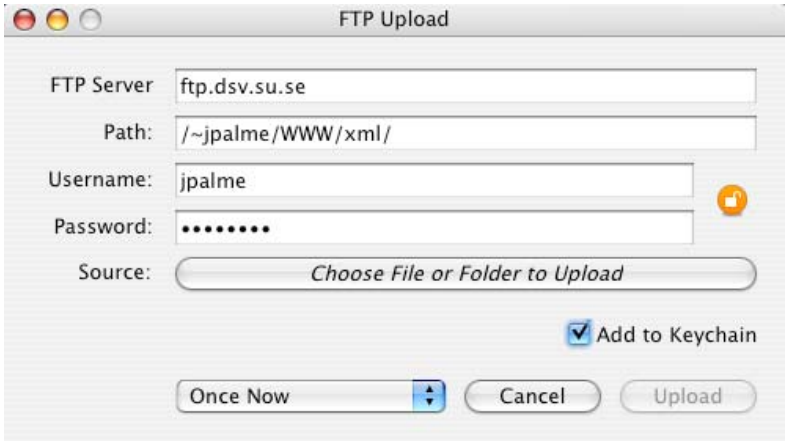
1. Combine the DTD and XML file into a single file as described in http://dsv.su.se/jpalme/internet-course/combine-dtd-xml.html
2. Paste the combined file into the window.
3. Click Validate to get it checked.

Text:
Suppress warning messages
Relax namespace checks
Validate Clear

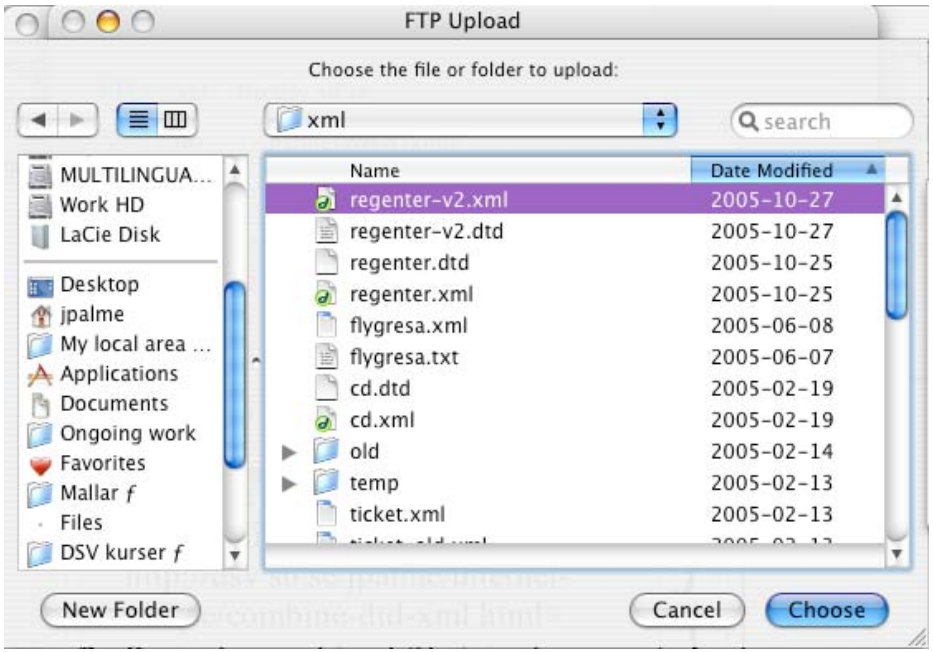
Uploading a file using an FTP/SFTP program:

This is difficult to describe in detail, because it depends on the user interface of the FTP/SFTP program. What is shown below is just one example with one particular FTP program and one way of using it.

Step 1: Start the FTP program and locate the area you want to upload the program to:



Step 2: Locate the file to upload:



Step 3: Confirm to start the upload: