Command Line shells

- Cmd on Windows
- Bash on Linux (and cygwin on Windows)

mosh@hmi-tech.net

Azularc, 4/2017

CMD Command line

Windows Disk consist of drives:

C:\ D:\ E:

And each disk has directories, e.g.

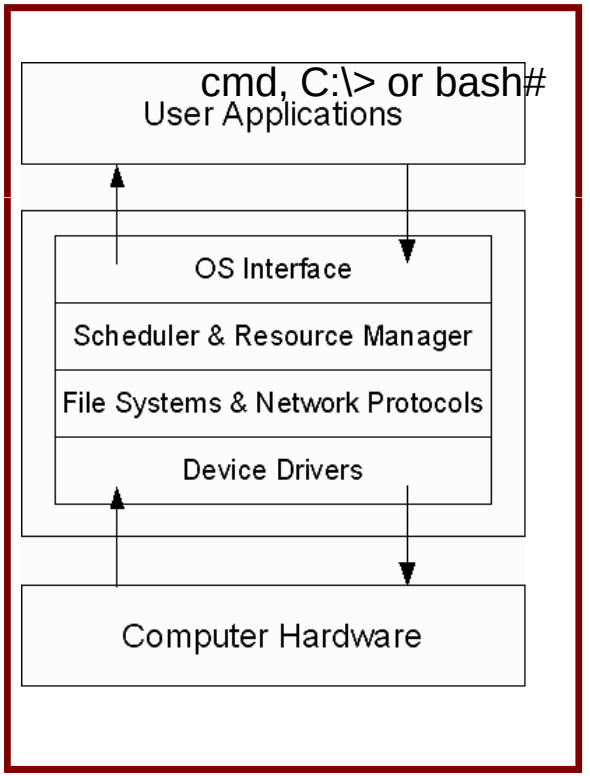
C:\windows **and**

C:\Documents and Settings\a\Desktop

And each directory contains files.

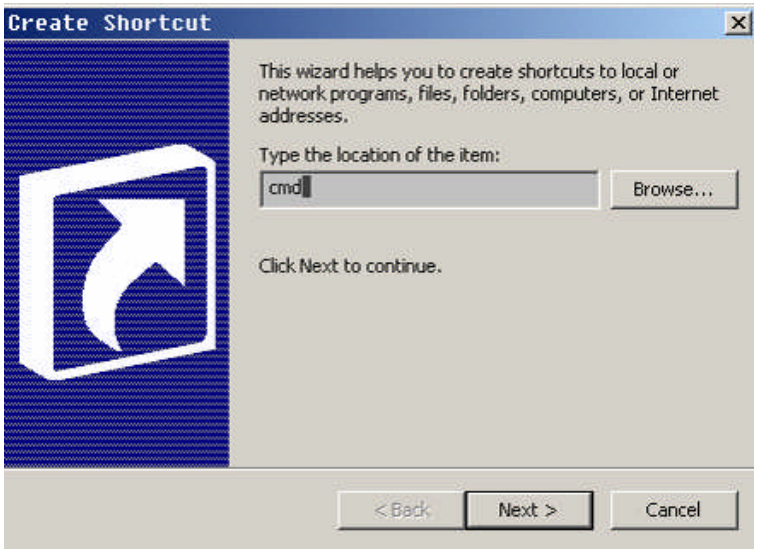
Files have a name and extension, e.g.

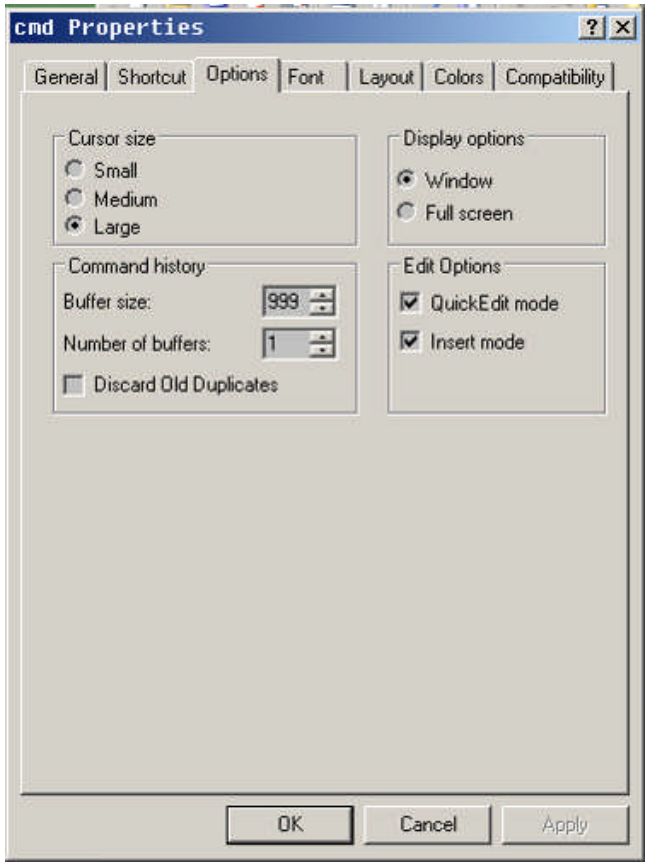
C:\WINDOWS\system32\system.ini



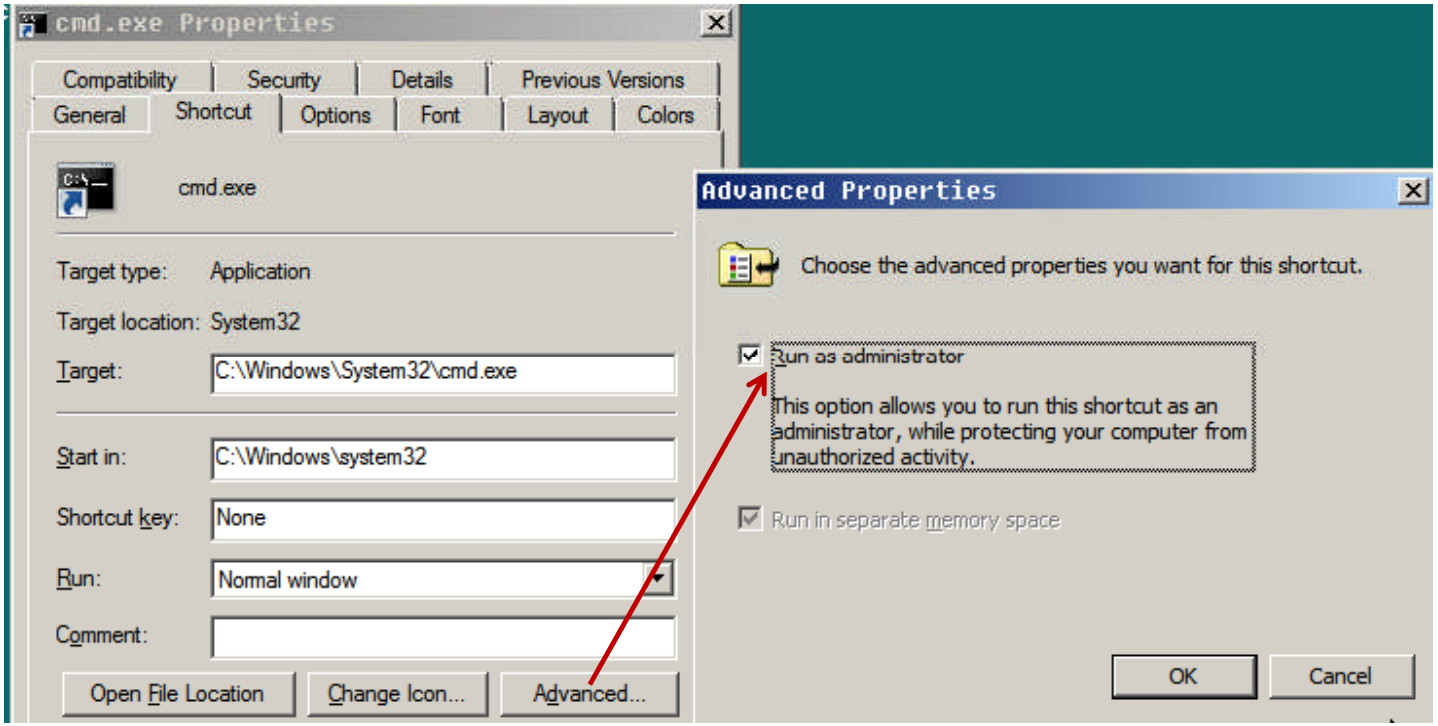
Create shortcut for cmd

- Right click -> new -> shortcut -> cmd
- Right click -> properties -> quick-edit



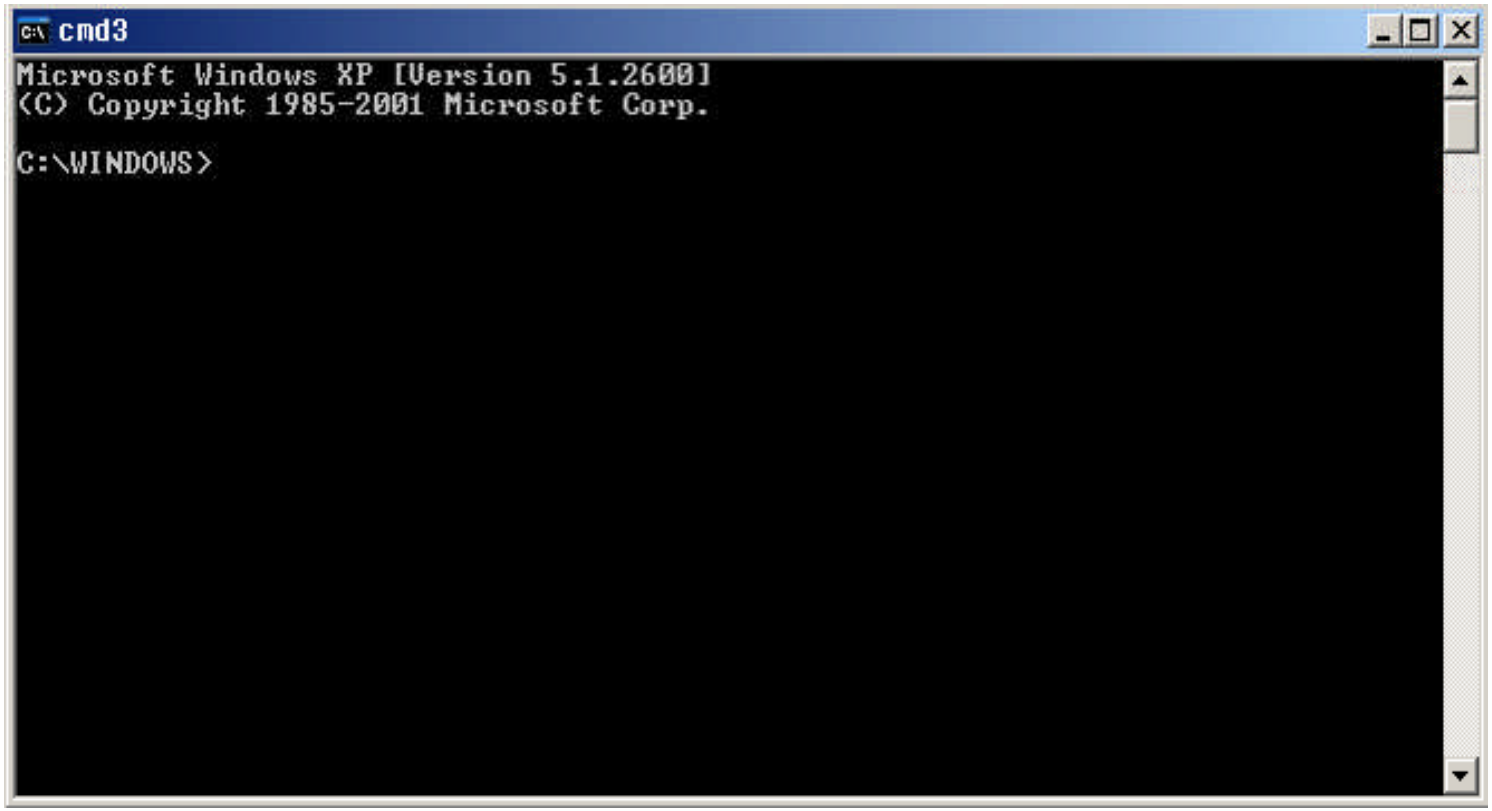


Always run cmd as admin



Cmd (console) window
Only text, no graphics

- Start -> Run -> cmd

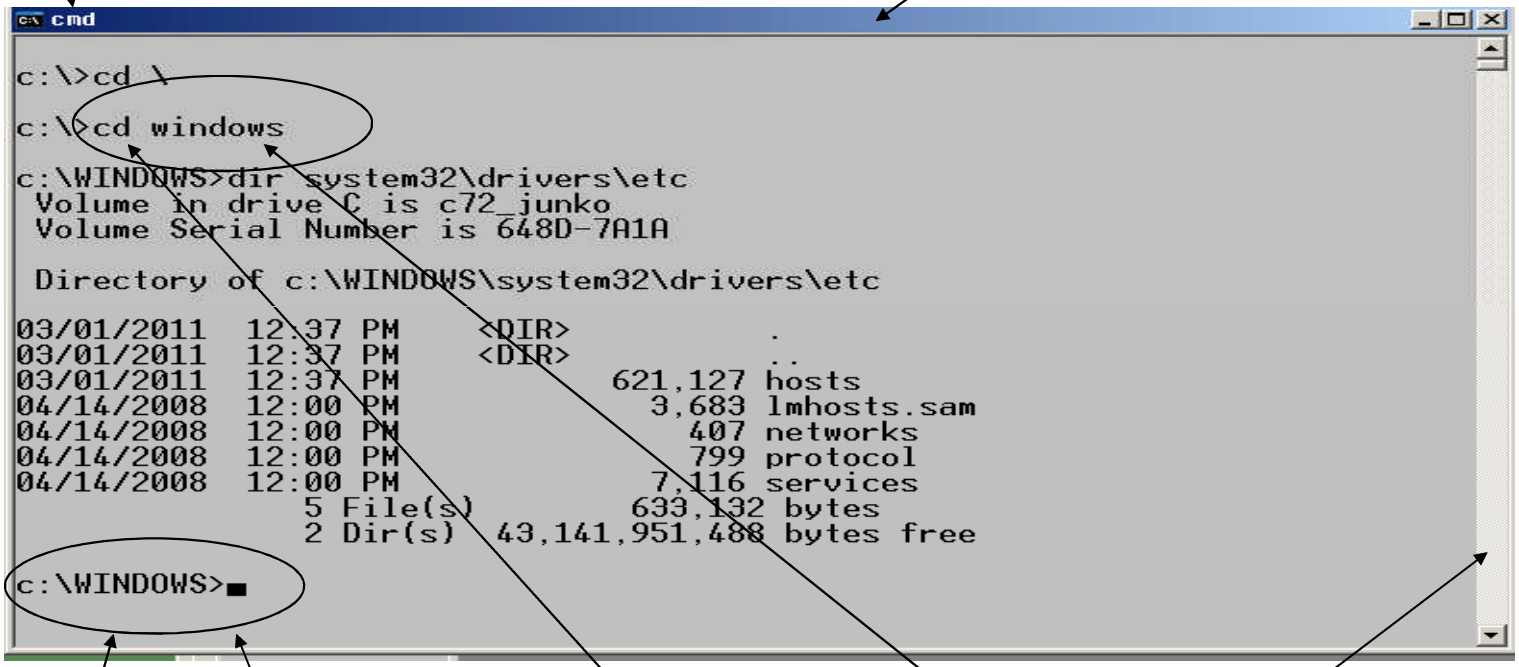


cd and dir commands

Cmd icon and name

Buttons to
- Minimize
[] Maximize
X Close window

Title bar



Prompt
Shows the current dir

Cursor to type next command

Command and argument

Scroll bar

Folders and files

Folder cabinets (File System)

C: ← Primary Disk volume

C:\ ← Root directory

C:\windows\ ← Windows Folder

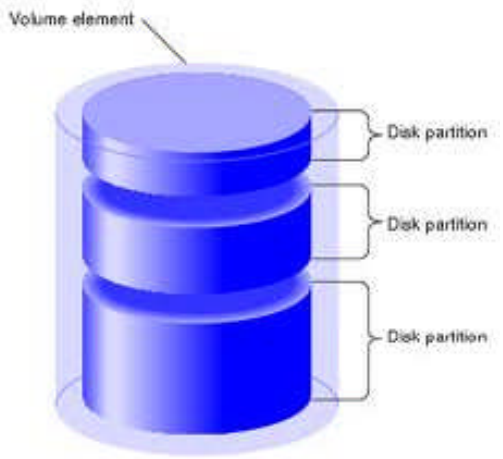
C:\Document and Settings\

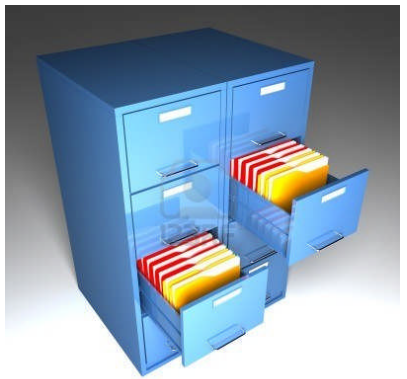
D: Secondary Disk

E: DVD drive

F: USB disk

Filesystems: Fat32, NTFS, ext2, etc

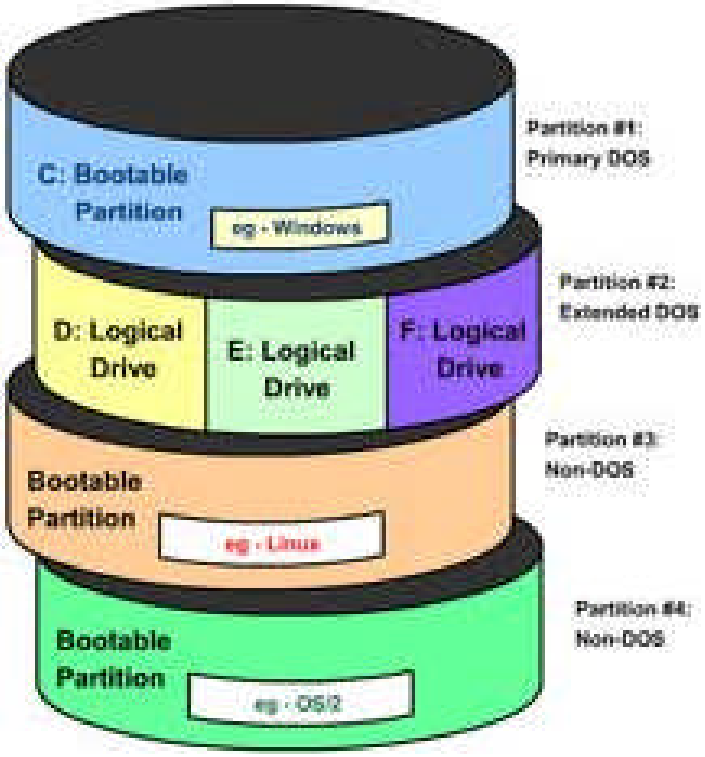




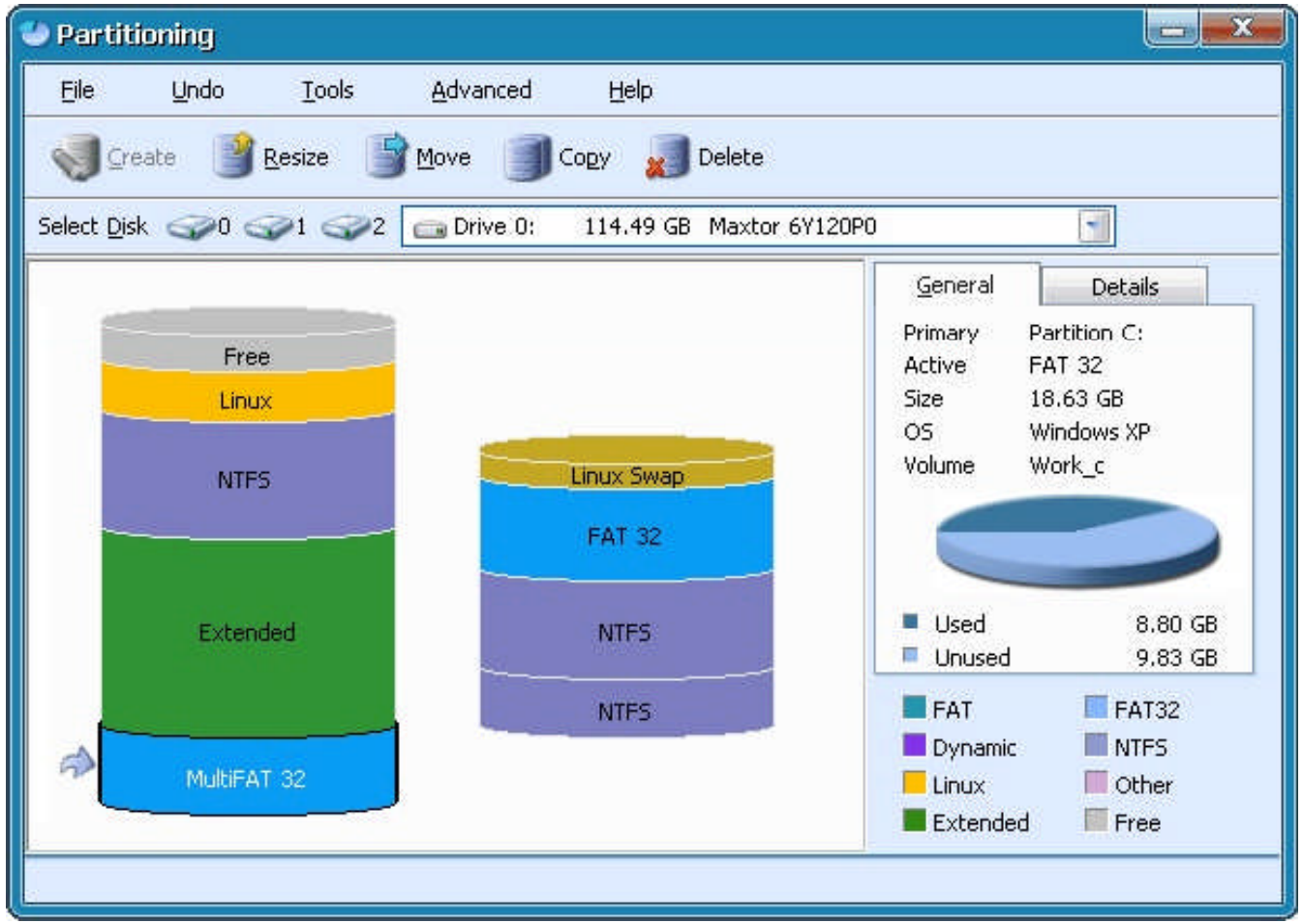
Disk volumes

/dev/hda (linux disk #1)

- Partition #1
  - C: boot partition (primary dos/windows)
- Partition #2 (ext dos)
  - D: logical drive, FAT32
  - E: logical drive, NTFS
  - F: logical drive, CDFS
- Partition #3, Non dos
  - Bootable, Linux
- Partition #4, non dos
  - OS/2 or BSD



Hard disk partitions



Filenames

- Files have a name and extension
- Avoid non-ascii chars in filenames
- Extension indicate type of file

  eg. readme.txt, photo.jpg, pic.gif, song.mp3, movie.avi

  Extensions can be wrong, e.g. photo.mp3

File attributes

Permissions:

- Readable, writable, readonly, execute
- Which users or groups can access it?

Timestamps:

- Time of creation, modified, access date

Size: in bytes

Type:

- text, binary, symbolic link
- stream, seekable, pipe, socket, lock

Paths

- Filesystems contain folders and files
- Folders and files have names
- Charset: Ascii, German, Unicode
- Paths are locations of Folders and files.
- Backslash is the DOS path separator
- Avoid non-ascii chars in paths, so it is easier to type in commands.
- Eg. C:\home\john is better than "C:\document and settings\john will"

Paths

Paths can be

- Absolute, example:
  C:\home\john\cc
- Relative to current folder
  . dot refers to the current folder
  .. dot-dot refers to the parent folder
  ..\.. Refers to the grand-parent folder
  Example: ..\readme.txt

Cmd keys

- Arrow-keys: command history editing
- F7 .. F8: command history
- TAB .. complete directory/filename
- Control-C (C-c) .. kill current command
- Control-Z (C-z) .. EOF (end of file).

Folder (directory) commands

C:\windows> cd ← Shows current dir

"C:\windows"

C:\windows> cd \ ← Change-dir to root

C:\> mkdir tmp ← Make dir C:\tmp

C:\> rmdir tmp ← Remove dir C:\tmp

Getting help

1. Google search
2. C:\> cd *\/?*

Displays the name of or changes the current directory.

CHDIR [/D] [drive:][path]
CHDIR [..]
CD [/D] [drive:][path]
CD [..] .. Specifies that you want to change to the parent directory.

Type CD drive: to display the current directory in the specified drive.
Type CD without parameters to display the current drive and directory.

Use the /D switch to change current drive in addition to changing current directory for a drive.
...

Command syntax

prompt> command **[/switches]**
[arguments]

Command completion, press [TAB]
repeatedly till the right word appears

C:\> cd C:\do<TAB><TAB>

C:\> cd c:\document and settings\<TAB>

C:\> cd c:\document and settings\john

File operations

C:\> Copy oldfile newfile

1 file(s) copied.

C:\> Copy oldfile directory

C:\> Rename oldfile newfile

C:\> Move oldfile folder

C:\> Delete oldname

Find dirs (folders)

C:\Documents and Settings> dir /s /b /ad goo*

C:\Documents and Settings\b\Application Data\Google
C:\Documents and Settings\Default User\Application Data\Google
C:\Documents and Settings\Default User\Local Settings\Application Data\Google

Options to commands:

Dir command takes following options (switches):

/s .. search Sub-directories also
/b .. just print Bare names
/ad .. result Attribute must be Directory (we don't want files)

Arguments to commands:

After switches, we type arguments to the dir command,
here we want folder names starting with goo..

Find files

Find files name ‘Hosts’ in C:\windows

```
C:\windows> dir /s/b hosts
    C:\WINDOWS\I386\HOSTS
    C:\WINDOWS\system32\drivers\etc\hosts
```

Search with wildcards:

```
C:\WINDOWS>  dir/s/b *socket*.*
   C:\WINDOWS\I386\MFSOCKET.IN_
   C:\WINDOWS\inf\mfsocket.inf
   C:\WINDOWS\inf\mfsocket.PNF
```

Saving output of commands

Search all doc files in c: drive and save the result to list-docs.txt file

```
C:\> dir /s/b *.doc > list-docs.txt
```

Save standard error output of gcc to a file

```
C:\> gcc bigfact.c 2> error.txt
```

Wildcards

Copy all files in . with .c extension to C:\tmp

```
C:\> copy  *.c   c:\tmp\
```

List all algo c file files:

```
C:\> dir algo*.c
```

List all algo files in any folder:

```
C:\> dir /s/b algo*.*
```

cmd wildcards to match filenames

- * matches any chars (zero or more)
- ? matches exactly one character

Examples:

- a*.? matches algo.c, algo.h, aah.c aaaa.c a.c
- a*b.d matches ab.d, axb.d, aabb.d,…
- *.* matches anything
- ?.? matches a.b, x.y, etc.

Search files, Regular expressions

Using GNU grep to find all c files containing the regular expression ‘Random…Numbers’:

C:\> grep –Pins random.*numbers *.c

- -P regular expression is perl syntax
- -i Ignore case,e.g. RANDOMNumbeR
- -n Print line number of match
- -s Ignore errors while searching

Regular (regexp) expressions

- Used to match strings in PERL, Python, C, Java, Vim, Emacs (text editor).
- Regexp syntax:

case sensitive

.	any one char
^	beginning of line
$	end of line
\n	newline
tom | jerry	tom OR jerry
“a(b | c)d”	grouping with parenthesis: abd or acd.
[a-zA-Z@#]	any char: a to z, A to Z, @, #.
[^a-z]	any char except a-z

Search output of a command

- C:\> dir | grep –Pi “(notes|exam)”
- Search the output of dir command for Notes or Exam, ignore case (perl regular expression)

- C:\> dir /s/b . | grep –Pi “(algo.*notes|number.*the.*oo)”
- Matches files names like “Algorithm Notes” and “Number-theory is good”.

Cmd Environment

Every process has an environment, it is a

Env is a list of variable=value, string pairs

C:\> set

SystemDrive=C:
SystemRoot=C:\WINDOWS
TEMP=C:\temp
USER=john

C:\> echo %TEMP%

TEMP=C:\temp

Beware of hidden spaces in env variable, e.g.

c:\> set tmp=c:\tmp<space>

c:\> rm –rf %tmp%/* .. this will delete everything: rm –rf c:\tmp<space>/*

Cmd aliases (shortcuts)

Make 'ls' mean the same as 'dir'

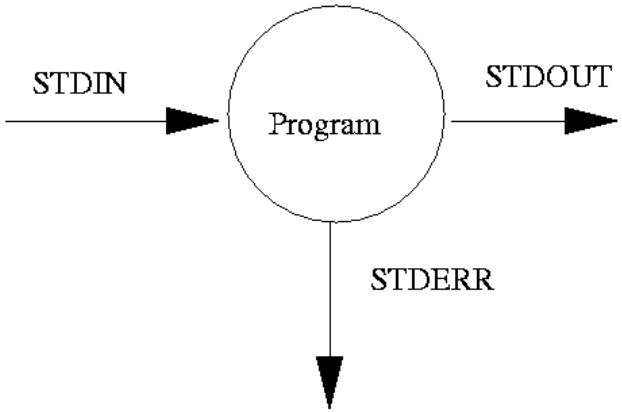
C:\> doskey ls=dir $*

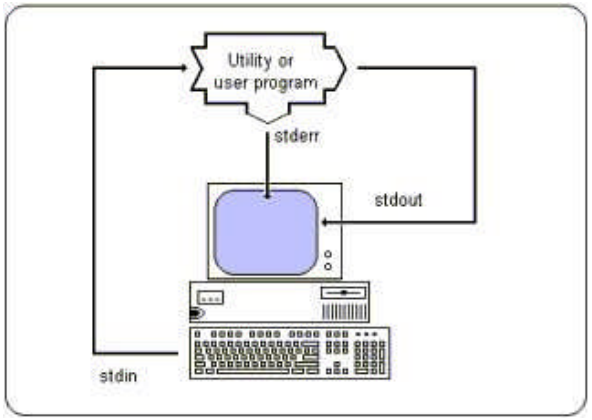
C:\> ls desktop.ini …. dir desktop.ini

IO Streams

Every process is connected to IO-streams:

0. stdin, standard input (keyboard).
1. stdout, standard output (monitor).
2. stderr, standard error (monitor).





PATH

How does cmd find the commands you type?

C:\> set PATH … Show the PATH

Path=C:\windows;C:\windows\system32;….

To Change the PATH

C:\> set path=C:\cygwin\bin;%PATH%

cmd batch files

```
C:\> more my.bat
@echo off
@rem this is a
   comment
echo Hello %USER%

C:\> my.bat
Hello john
```

```
C:\> more my.cmd
::  A comment
echo Hello %USER%



C:\> my.cmd
Hello john
```

Cmd Tricks and Tips

1. Use *Clink* for Bash style completion; and ConEmu for multi tab consoles.

   (google search for clink and conemu cmd)

2. Calling external commands from a cmd script

```
:: Set MYDATE=2016-09-09 by calling gnu date
for /f "delims=" %%a in ('c:/cygwin/bin/date +%%Y-%%m-%%d') do @set MYDATE=%%a
```

3. Use cygwin / bash / perl for more complicated scripting.

Cmd Tips - mklink

```
> mklink /d c:\Desktop  "c:\Users\%USERNAME%\Desktop"

> mklink  /d  c:\download
    "c:\Users\%USERNAME%\Documents\Downloads"

> mklink  /d  c:\SendTo
    "c:\Users\%USERNAME%\AppData\Roaming\Microsoft\
    Windows\SendTo"

> mklink hosts.txt  %WINDIR%\system32\drivers\etc\hosts
> mklink  /d  c:\Pf32 "c:\Program Files (x86)"
> mklink  /d  c:\Pf64 "c:\Program Files"
> mklink /d   c:\e\      e:\
# Now you can access /e/file without colons
```

Useful Tools on Windows

1. Cygwin, bash, perl, gcc
2. Notepad++, gvim
3. Google Chrome
4. Process Exporer, tcpview
5. Java OpenJDK (not Oracle-Java)
6. Xampp (web server)
7. Python 2.7

8. IrfanView,VLC,1by1
9. CodeBlocks

Bash shell

mosh@hmi-tech.net

Azularc, 4/2017

Bash shell

Default terminal shell (command interpreter) on Linux is /bin/bash

Windows c:/cygwin/bin/bash or c:/cygwin64/bin/bash

History:

sh (ATT unix) → ksh → bash (current)

Obselete: zsh, csh, tcsh (bad designs)

Windows Cygwin Setup

- Google > download Cygwin > setup*.exe
- Run cygwin setup.exe > next > next …
- Installs in c:/cygwin64 or c:/cygwin
- start > run > [cmd as admin]

> set PATH="c:\cygwin64\bin;%path%"

Save PATH in registry for all users

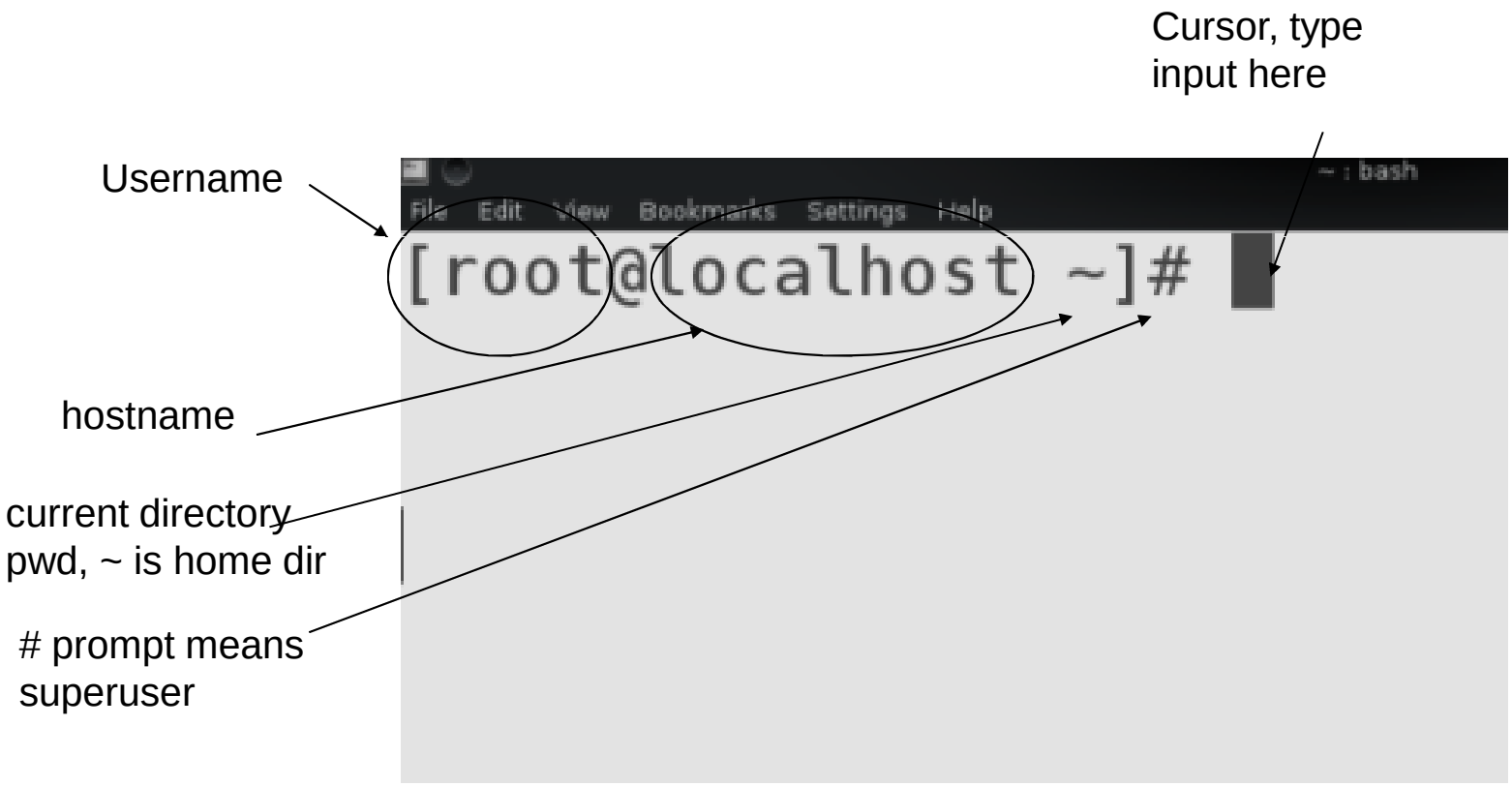
> setx PATH "c:\cygwin64\bin;%path%" –m

> bash

$

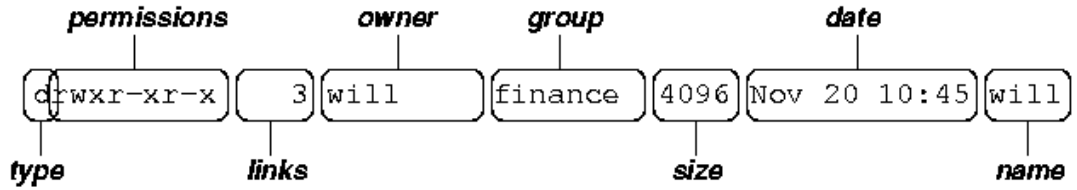
Start bash in a terminal

- start > terminal



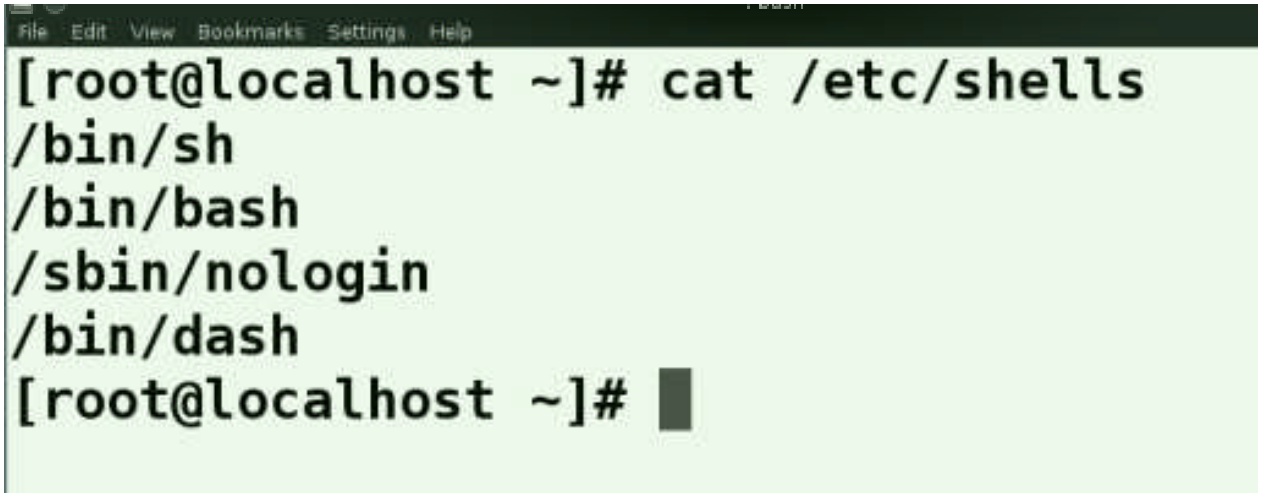
ls – list directory and files

bash$ ls –l will .. –l option for long details



cat (concatenate, print file)

cat filename .. prints the filename to stdout, which is the terminal screen, also called /dev/tty.



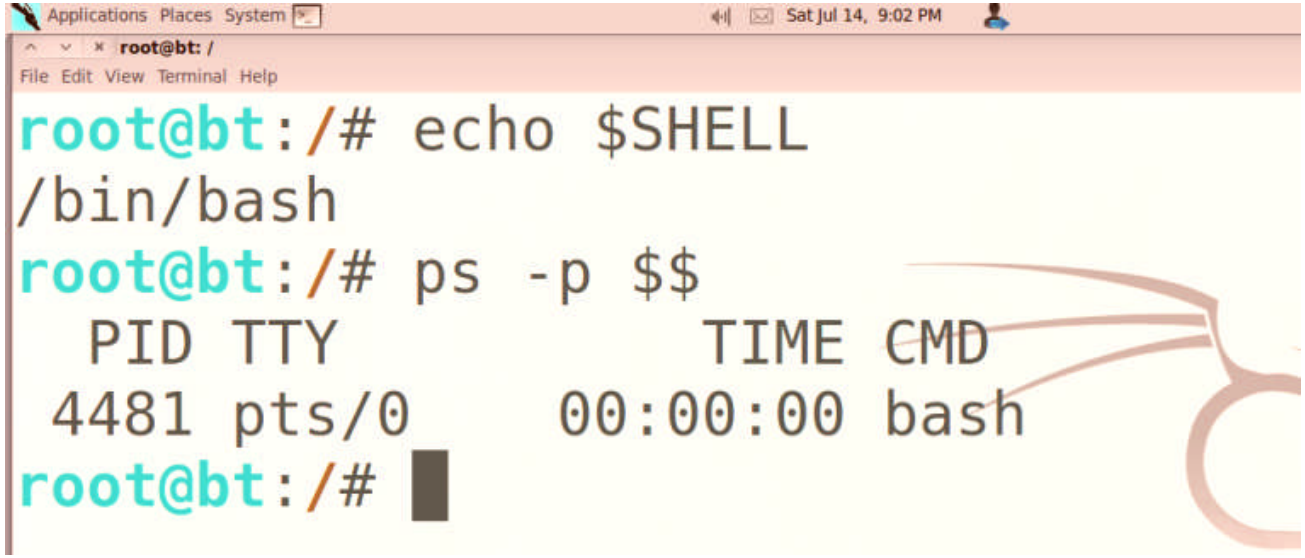
Environment

echo $HOME

ps -p $$ .. process info

$$ .. Is the pid (process id) of this shell.

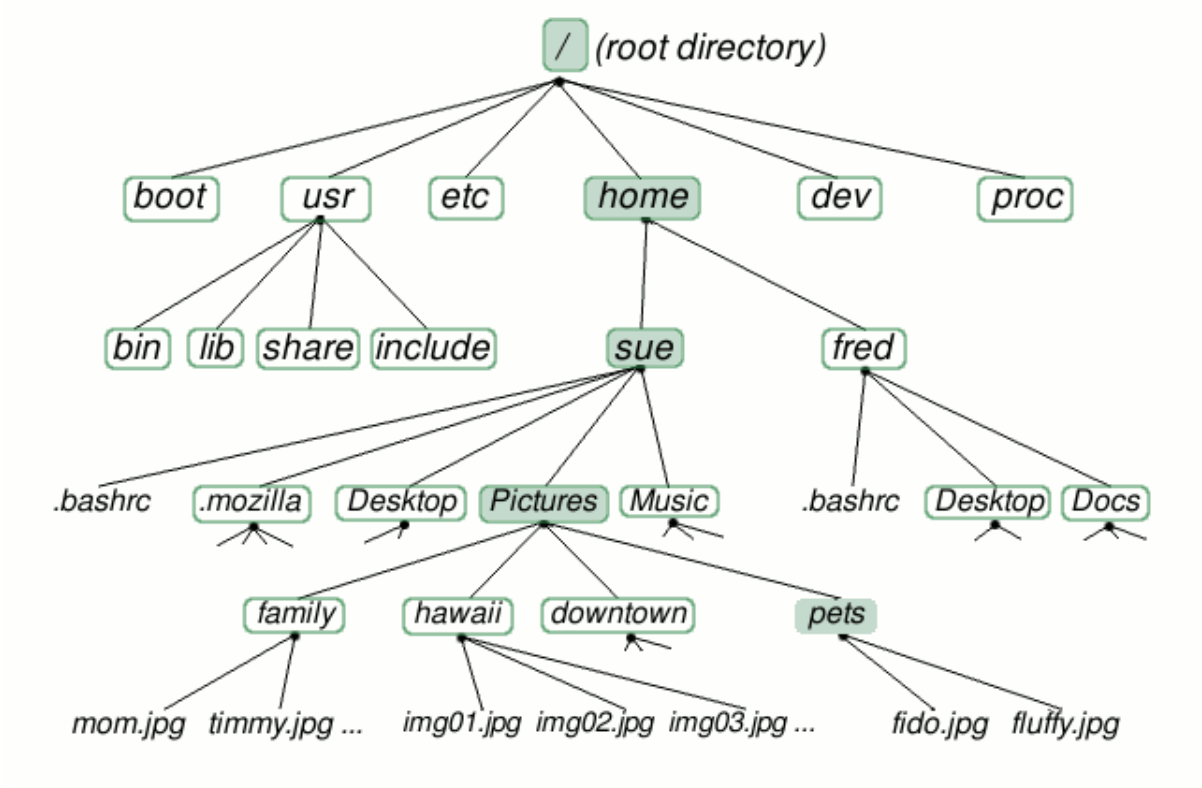
TTY .. is the controlling terminal of this bash



Unix filenames

- / is the root
- /dev/ .. are the devices, like keyboard, tty, harddisks.
- /bin .. are the programs
- /home/john .. user directory
- /etc .. system configuration (like registry)
- /usr .. user applications
- Symlinks, one link can point to another

Unix file system



Common terminal keys

- Control-Z .. suspend command
  - fg .. restart command in foreground
  - bg .. send command into background
- Control-C .. interrupt current command
- Control-\ .. Kill current command
- Control-D .. EOF to logout
- Control-S .. Stop screen output
- Control-Q .. continue screen output.

Readline (Command line editing) in bash

- C-a .. beginning of line
- C-e .. end of line
- C-r .. search history
- Up-arrow .. previous history command
- Down-arrow .. next history commands
- C-k .. delete to end of line

See google, same as Emacs editor keys, can remap keys in ~/.inputrc

Common Unix commands

- ls files .. list file or directory
- cat files .. print files to stdout
- man xyz .. show manual help page for xyz
- cp source target .. copy source to target
- mv source target .. move
- rm source .. remove source
- cd /usr/local .. change directory to
- pwd .. show present working dir
- grep regexp file .. search regexp in files
- more files .. show files page by page.

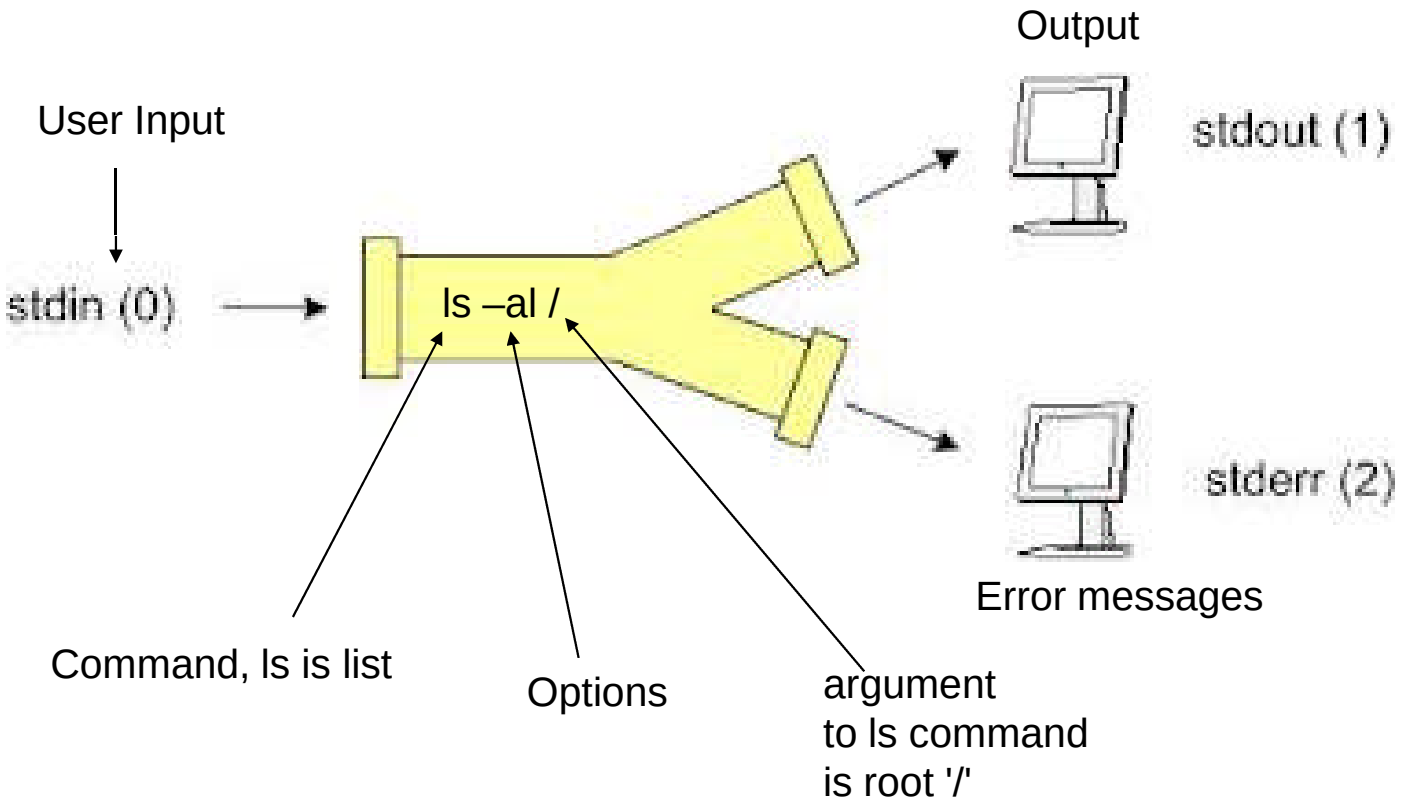
More Unix commands

- ps … show processes
- who … show who is logged on
- date .. show date
- cal .. show calendar
- stty .. terminal settings
- chmod .. change dir/file permissions
- vim files .. vi improved editor
- emacs files .. emacs editor

Network commands

- ping host .. check network connection to host
- tracert host .. trace route to host
- nslookup .. DNS name lookup
- mail .. read or send email
- ftp .. file transfer
- wget urls .. download urls
- telnet host .. login to host
- ssh host .. secure shell login to host
- finger user@host .. find out about user on host

Process and its IO



Saving output to a file

Count number of lines in /etc/shells and save it to x

```
$ wc –l /etc/shells  > x
$ cat x
   16 /etc/shells      .. number of lines in file
```

Save errors to a file (stderror is fd2):

```
$ gcc –Wall bigfact.c 2> errors.txt
$ more  errors.txt  ... show the file page by page
```

Reading input from a file

```
$ wc –l <  /etc/shells
   16 lines
```

Redirect input and output

```
$ wc –l < /etc/shells > x
```

Saving outputs

Save output, redirect stdout to a file.

```
$ wc /etc/shells > /tmp/y
$ cat /tmp/y
  16  16 186 /etc/shells
```

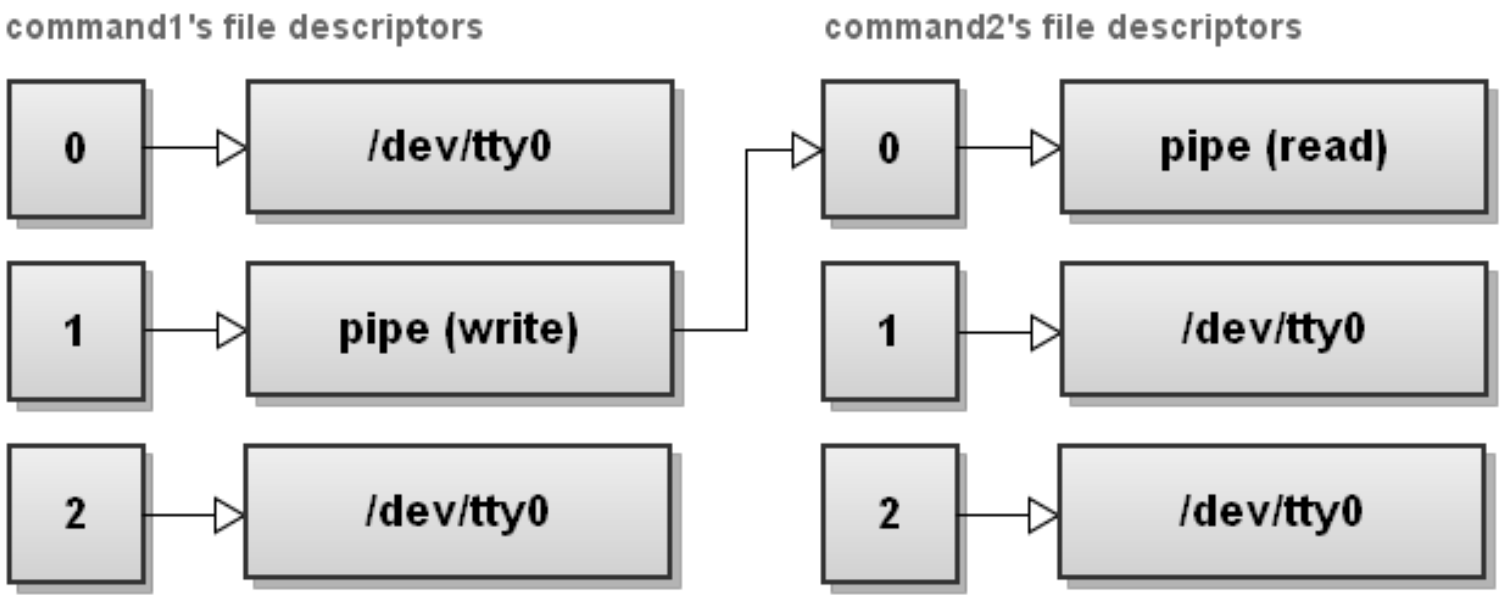
(means 16 lines,16 words, 186 chars in /etc/shells)

Save output and error messages of gcc, send stdout to file x, and also redirect stderr/2 to stdout/1.

```
$ gcc -Wall bigfac.c  >  x     2>&1
```

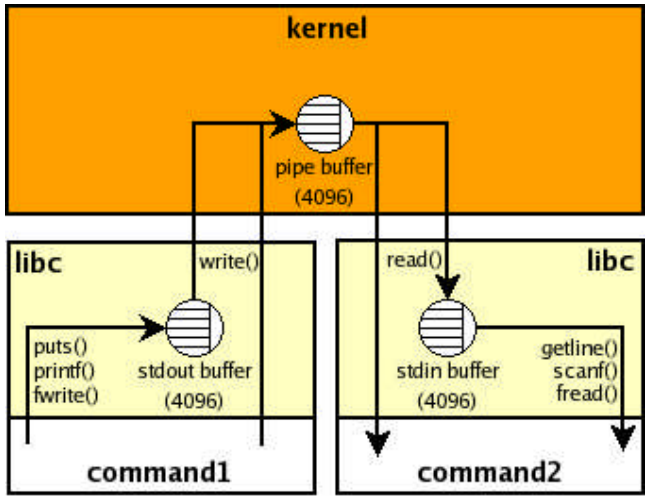
pipe, and io redirection

```
$ command1 | command2
```



Piping ‘|’

Pipe output of first cmd to next cmd, example

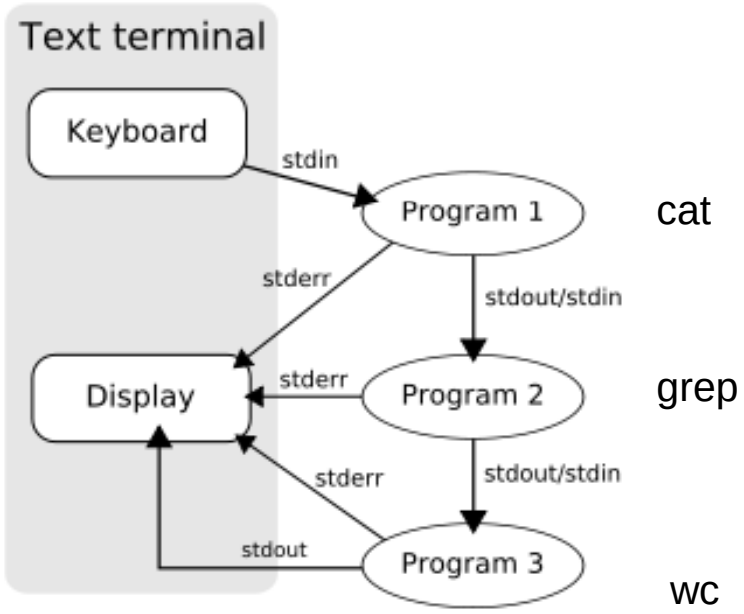



```
$ cat /etc/shells | wc
   16 16 186
```

Pipe example with 3 commands

Example: Count number of lines in file containing the string 'sh' or 'SH' ..

```
$ cat /etc/shells |
  grep -i sh |
  wc -l
2
```



Running cmd in background

```
$ wc /etc/shells > /tmp/x  &
  [1] 2804          ... process number of background job.
$ ls
  [1]+ Done ... later background job is done
```

Quoting arguments

```
$ echo *            .. ‘*’ is globbed into matches
home etc usr
$ echo “*”          .. prevent globbing of *.
*
$ echo \*           .. backslash quotes next char
*
```

Quoting Variables

```
$ echo $HOME
   /home/john
$ echo 22${HOME}99
  22/home/john99

$ echo ‘$HOME’
   $HOME
$ echo \$HOME
  $HOME
```

Bash aliases

Make 'dir' same as 'ls –al' command

```
$ alias dir='ls –al'
$ alias   date-is='date +%Y-%m-%d'
$ date-is
2013-04-13
```

Bash functions

```
$ function dates(){
    echo DATE_SECONDS=$(perl -e "print time")
    echo DATE_YESTERDAY=$(date --date="1 days ago" +%Y-%m-%d)
    echo DATE_TODAY=$(date --date="0 days ago" +%Y-%m-%d)
    echo DATE_TOMORROW=$(date --date="1 days" +%Y-%m-%d)
}
$ dates
DATE_SECONDS=1365864924
DATE_YESTERDAY=2013-04-12
DATE_TODAY=2013-04-13
DATE_TOMORROW=2013-04-14
```

bash scripting

```
$ cat script
   #!/bin/bash
   # my first comment in this file.
   echo “My first script, hello $USER”
$ chmod +x script
$ ./script
   My first script, hello john
$ bash –x –v script      .. To debug verbose
   My first script, hello john
```

Bash script commands, if then

```
$ cat myscript1.sh      # comment.
if [[ file1 –nt file2 ]] ;then
    echo “file1 is newer”
;elsif [[ 20 –gt 5 ]] ;then
    echo “20 is greater than 5”
;else
    true; # dummy stmt.
; fi
```

case stmt

```
$ cat myscript2.sh
case $# in
0) echo You typed no arguments ;;
1) echo You typed $1 ;;
2) echo You typed $1 and $2 ;;
*) echo You typed $* ;;
esac
```

for loop

```
$ for user in a b c ;do
      ls –l /home/$user
; done > list.txt
```

while loop

```
$ file=/tmp/x.log
$ while [[ ! –s $file ]] ; do
   echo waiting for $file to fill up
   sleep 1
   done
```

Perl power user

Fix spelling of 'thier' to 'their' in all c files

```
$ perl -p -i.bak -e 's/\bthier\b/their/g'  *.c
```

---

s/// is Substitute/search-regexp/replacement/

Options:

-p print .. print each line after substitute

-i.bak .. save original as file.bak

-e expr .. to execute perl expression on each line

Windows / Unix differences

	Dos/Windows	Unix/Linux/Bsd
File separator	****	**/**
Root	C:\	**/**
Line ending (Fix dos2unix, unix2dos)	\r\n	\n
Shell	cmd	bash
File case	dir == DIR	ls != LS
Syntax	Inconsistent	Perfect
variables	%USER%	$USER

Broken windows commands with same names as Unix commands:

- echo
- find
- date
- time
- for
- if
- mkdir
- link

Other tools:

- putty (windows connect to unix)
- ssh (secure shell)
- rsh (restricted shell)

HTML

Definitions and Acronyms

- HTML - hyper text markup language
- DOM – Document Object Model
- CSS - cascading style sheet
- JS - javascript
- URL - universal resource locator
- HTTP - hyper text transfer protocol
- HTTPS - http secure
- IP - internet protocol

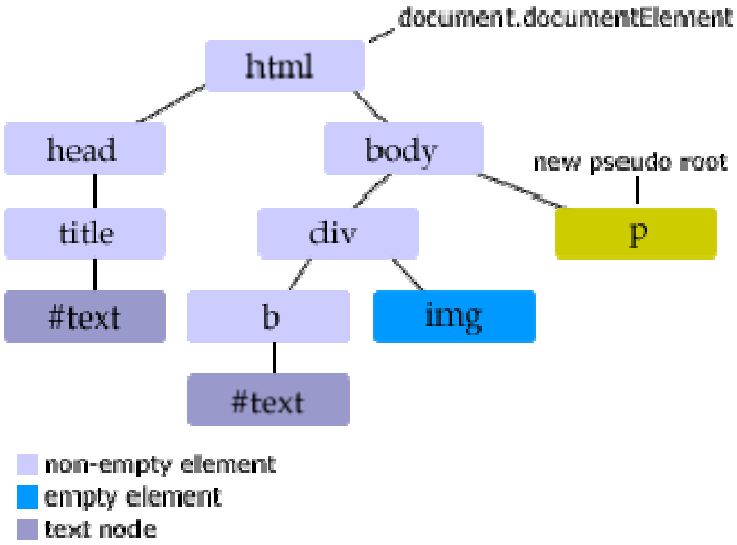
Making a Webpage DOM

Dom is made of

- <tag>
- </tag> .. closing tag
- <tag attribute=value>
- elements

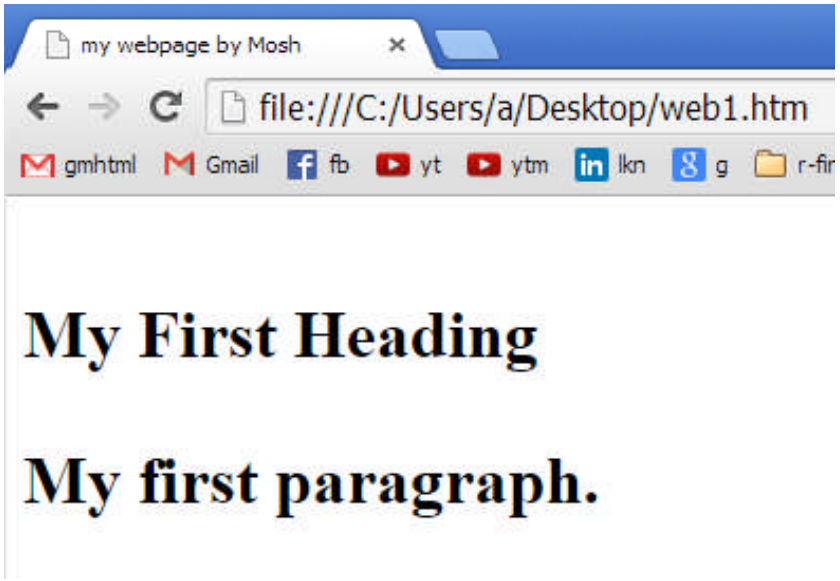
webpage.html with notepad++

```
<!DOCTYPE html>
<html>
<head>
    <title> webpage by NAME
   </title>
</head>
   <body>
    <h1> Heading </h1>
    <p>  paragraph 1
    <p> Para 2
   </body>
</html>
```



Open the file in Chrome

- Notepad > file > save as > file=“web1.htm”
- Chrome > Control-O (open) > “web1.htm”



Tags

<body> -- tag begin

more NODEs between tags

</body> -- /tag end

Some tags are self closing:

 or
, <P>

<img src="file.jpg">

Comments

< ! - -

COMMENT not displayed

- - >

Headings.htm

<h1> HEADING one </h1>

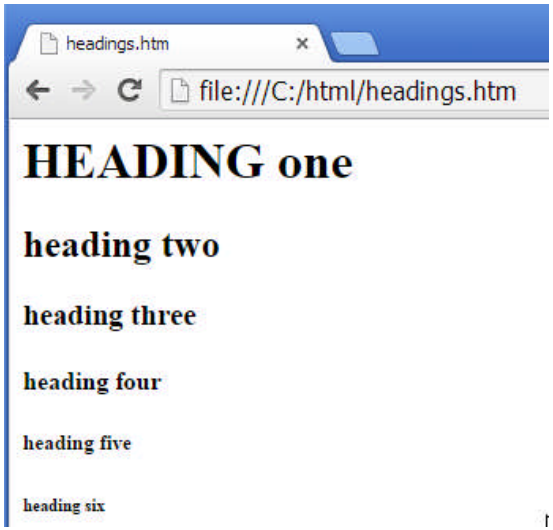
<h2> heading two </h2>

<h3> heading three </h3>

<h4> heading four </h4>

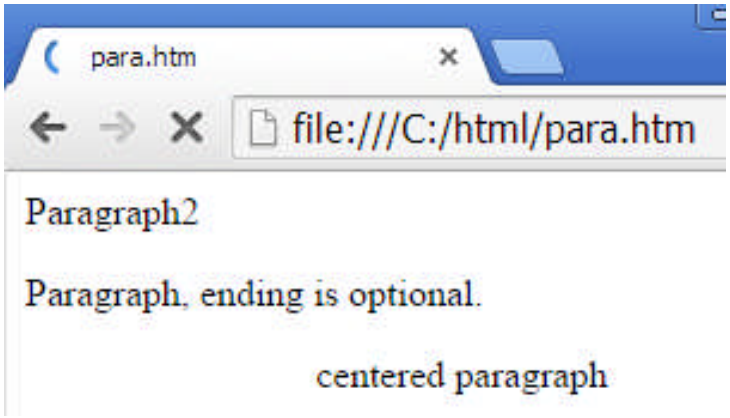
<h5> heading five </h5>

<h6> heading six </h6>



para.htm

```
<p>  Paragraph2 </p>
<p>  Paragraph, ending is optional.
<p align=CENTER>  centered paragraph
 </p>
```

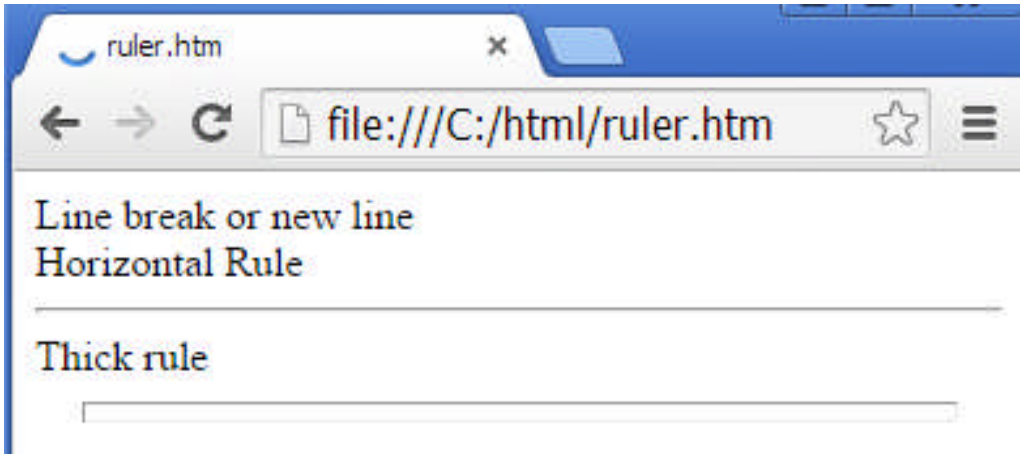


Ruler.htm

Line break or new line

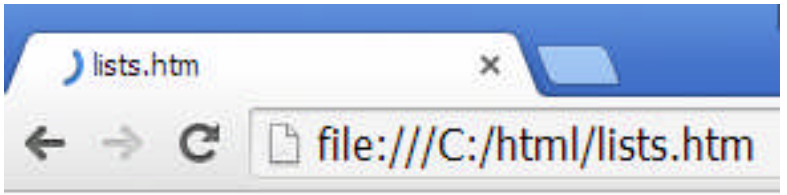
Horizontal Rule <hr>

Thick rule <hr size=8 width="90%">



Lists

```
<ul>   UNNUMBERED LISTED.
  <li> Item A
  <li> Item B
</ul>
<!-- ------------------- -->
<ol>   NUMBERED LIST.
  <li> Item A
  <li> Item B
</ol>
<!-- ------------------- -->
<dl>   DEFINITION LIST.
  <dt> TAG1 <dd> DATA
  <dt> TAG2 <dd> DATA
</dl>
```



UNNUMBERED LISTED.

- Item A
- Item B

NUMBERED LIST.

1. Item A
2. Item B

DEFINITION LIST.
TAG1
DATA
TAG2
DATA

Anchors

HREF

Description of URL

link to image

 EMAIL

Marks in URL

 Name of MARK1

Click to JUMP to Mark1

Link to MARK1 in URL

Images

```
<img src="kailash.jpg">

<a href="forest.jpg">
click to see image
</a>

<img src="kailash.jpg"  alt=“kailash photo">

<p align=“center”>
   centered Image
</p>
```

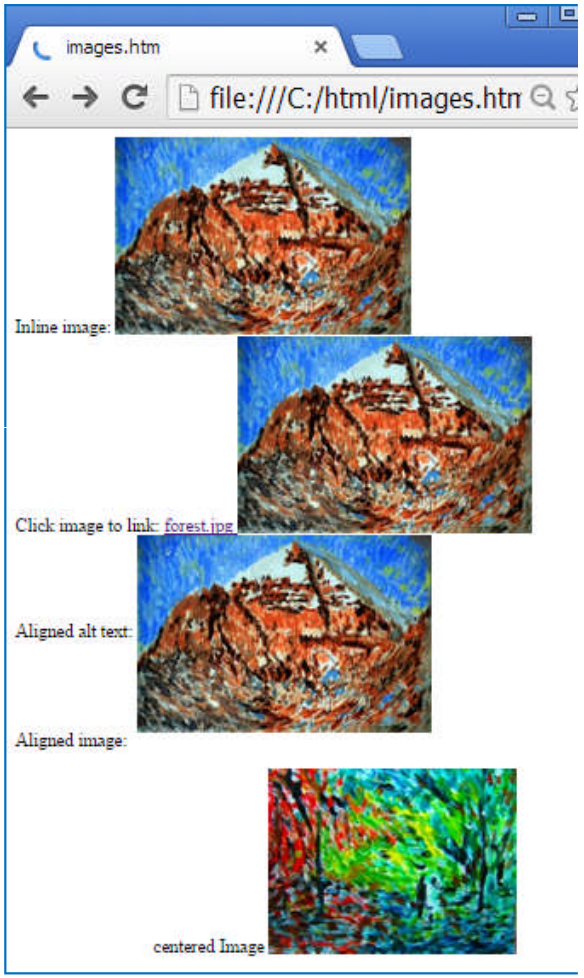
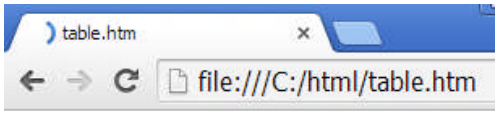


Table.htm

```
<table border="1" cellpadding="1" cellspacing="1"
   summary="table2">
    <caption> Food cost table </caption>
    <tr> <th> Food </th>     <th> Cost </th> </tr>
    <tr> <td> Rice </td>     <td>  $11 </td> </tr>
    <tr> <td> Dal  </td>     <td>  $12 </td> </tr>
  </table>
```

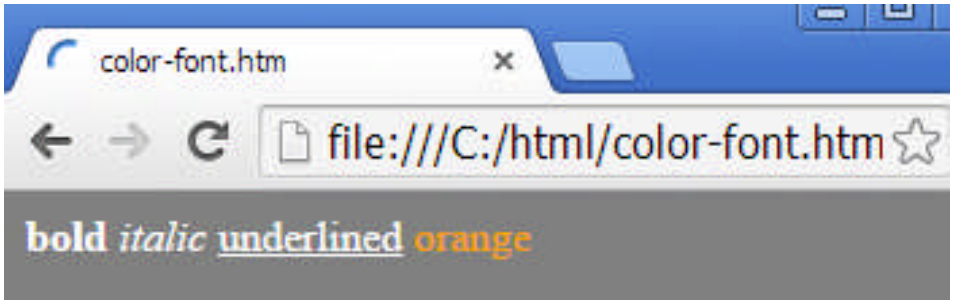


Food cost table

Food	Cost
Rice	$11
Dal	$12

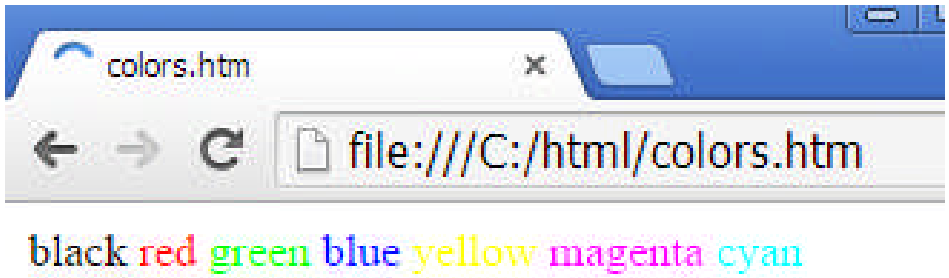
color-font.htm

```
<BODY   BGCOLOR="GRAY" TEXT="WHITE"
  LINK="YELLOW"  VLINK="YELLOW" >
    <B>bold</B>
    <I>italic</I>
    <U>underlined</U>
    <font color="orange"> orange</font>
</BODY>
```



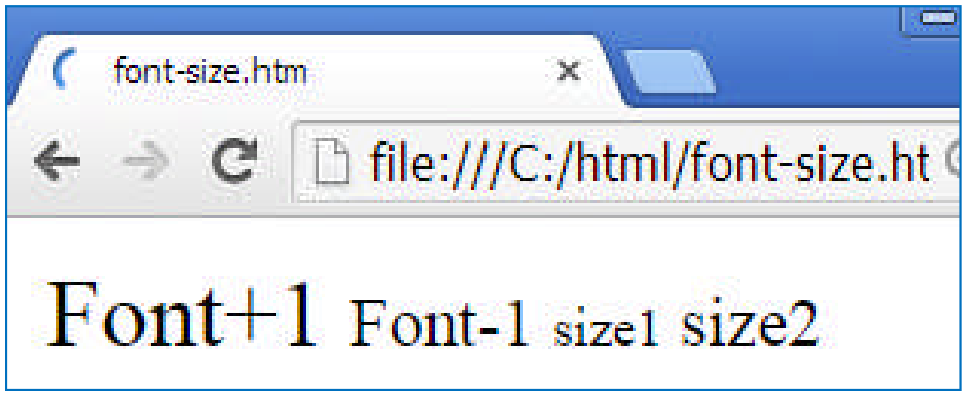
Colors.htm

```
<FONT color="#000000">black</FONT>
 <FONT color="#FF0000">red</FONT>
 <FONT color="#00FF00">green</FONT>
 <FONT color="#0000FF">blue</FONT>
 <FONT color="#FFFF00">yellow</FONT>
 <FONT color="#FF00FF">magenta</FONT>
 <FONT color="#00FFFF">cyan</FONT>
 <FONT color="#FFFFFF">white</FONT>
```



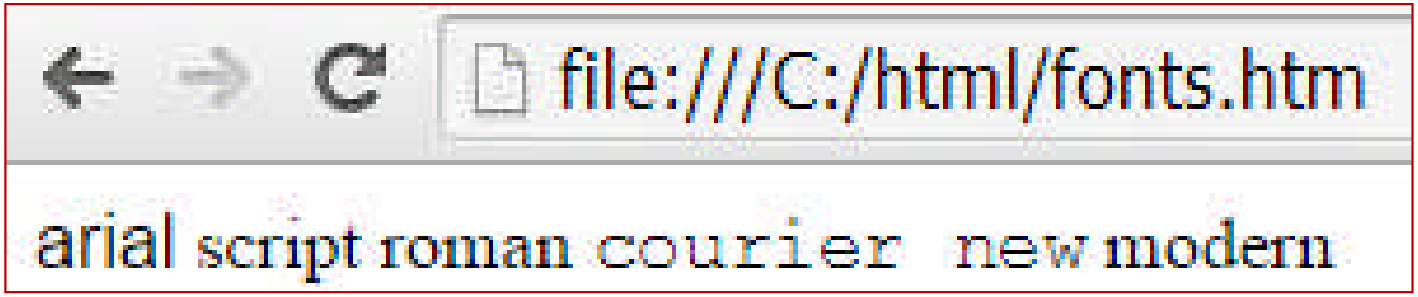
font-size.htm

```
<FONT SIZE="+1">Font+1
<FONT SIZE="-1">Font-1
<FONT size=1>size1</FONT>
<FONT size=2>size2</FONT>
```



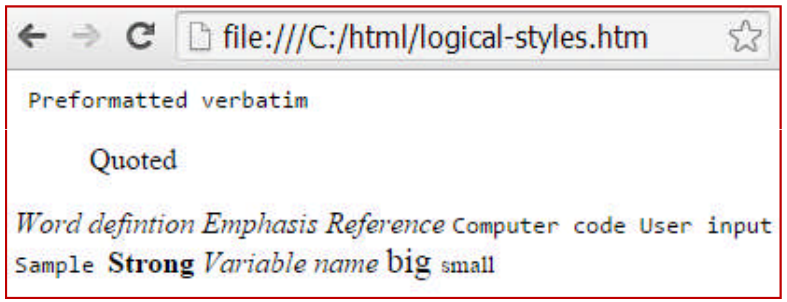
Fonts

```
<FONT face="arial">arial</FONT>
 <FONT face="script">script</FONT>
 <FONT face="roman">roman</FONT>
 <FONT face="courier new">courier
  new</FONT>
 <FONT face="modern">modern</FONT>
```



logical-styles.htm

<pre> Preformatted verbatim </pre>
<blockquote> Quoted
</blockquote>
<dfn> Word defintion </dfn>
 Emphasis
<cite> Reference </cite>
<code> Computer code </code>
<kbd> User input </kbd>
<samp> Sample </samp>
 Strong
<var> Variable name </var>
<BIG> big </BIG>
<SMALL> small </SMALL>



physical-style.htm

 Bold

 Italic

<tt> Typewriter </tt>

<I><B> bold and italics</B></I>

<U> underline </U>

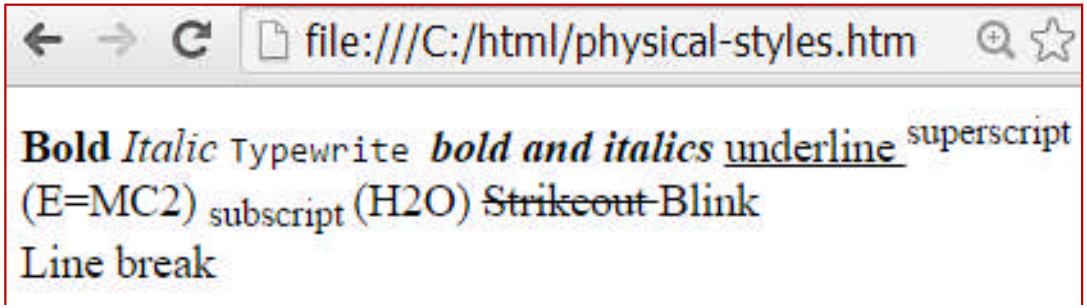
^{superscript}($E=MC^2$)

_{subscript} (H_2O)

<S> Strikeout </S>

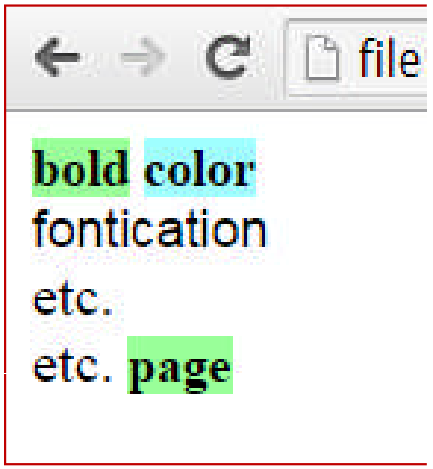
<BLINK> Blink </BLINK>

 Line break



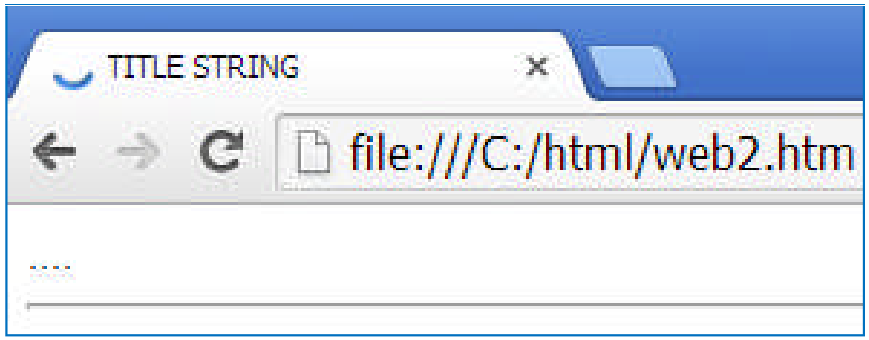
Inline-Style.htm

```
<b style="color: black; background-color:
    rgb(153, 255, 153);">bold</b>
<b style="color: black; background-color:
    rgb(160, 255, 255);">color</b>
<div style="font-family: Arial,Helvetica,sans-
    serif;">fontication</div>
<div id="content" style="width: 750px; border-
    right: medium none;">
<span style="font-size:
    18px;">etc.<br>etc.</span>
<span style="background: none repeat scroll
    0% 0% rgb(153, 255, 153); color:
  black; font-weight: bold;">page</span>
```

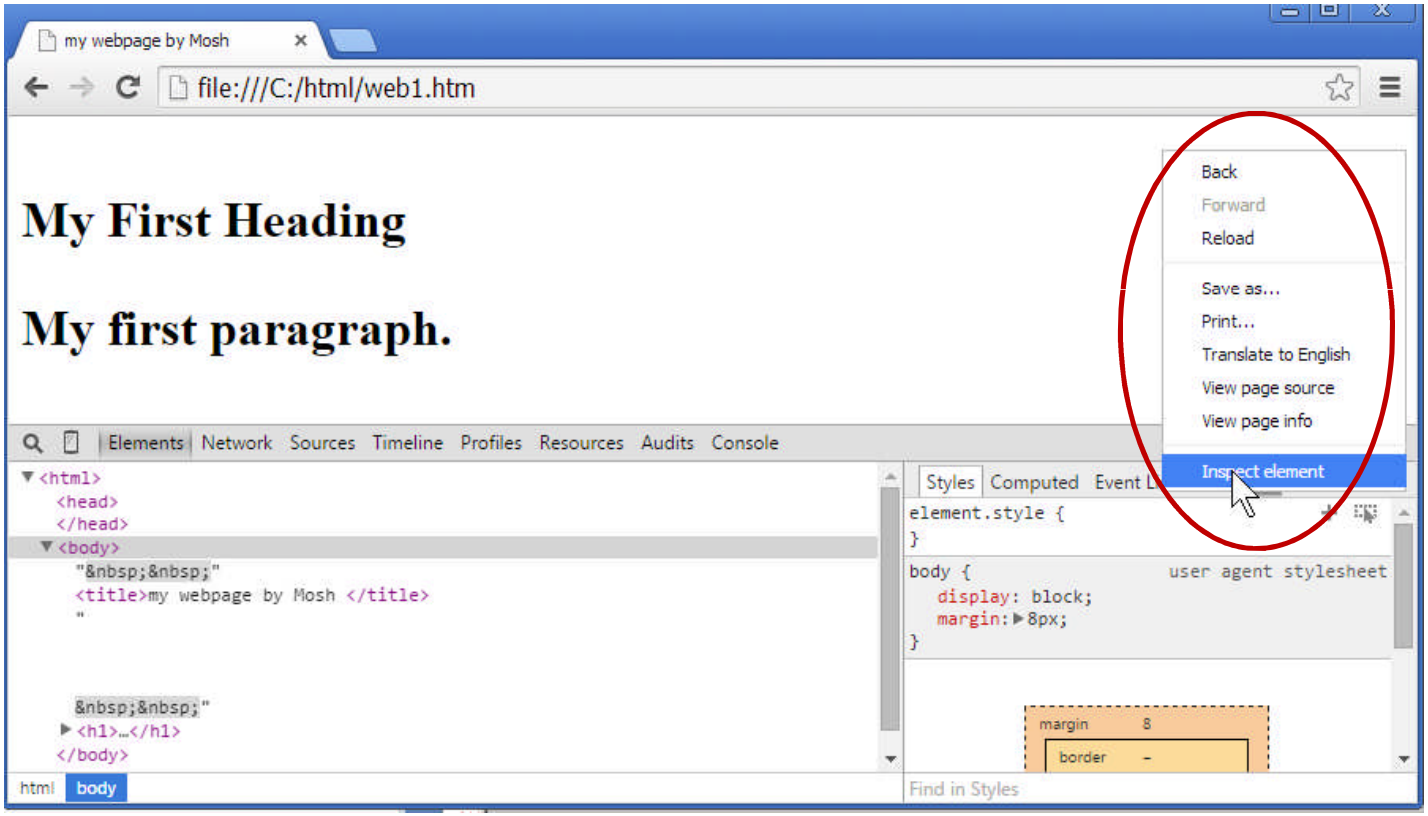


web2.htm

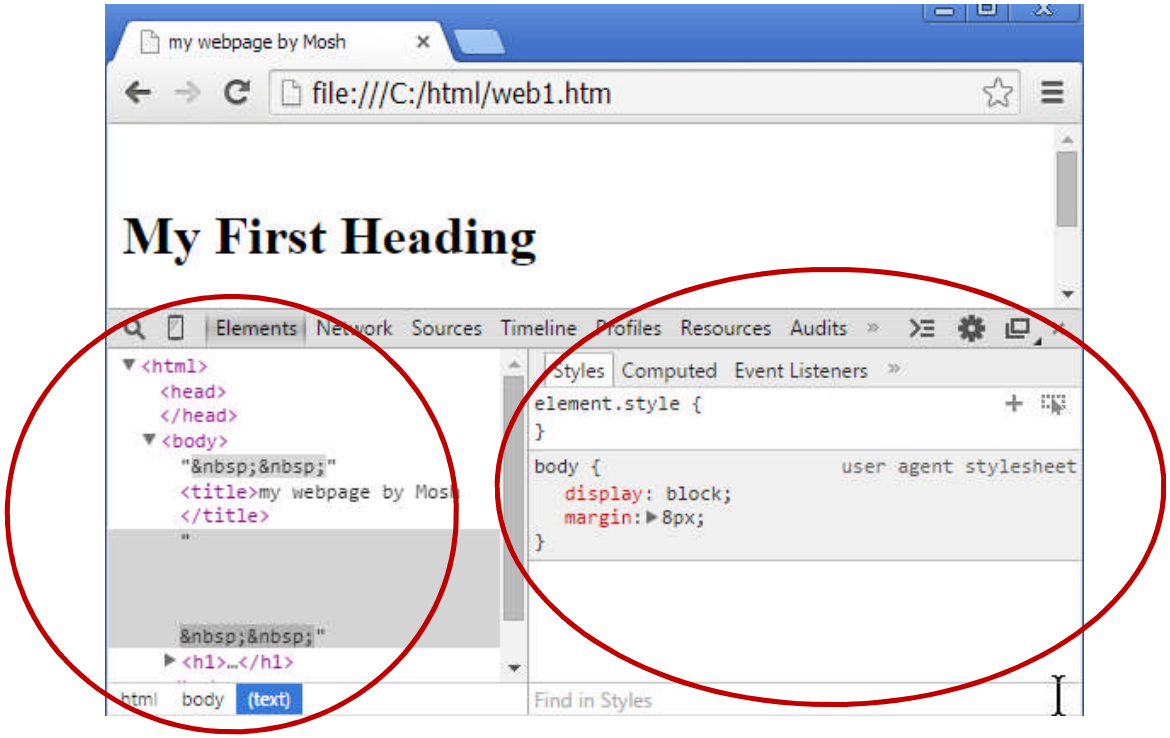
```
<base href="URL">
<html>
  <head>
  <title> TITLE STRING </title>
  </head>
  ....
  <!-- COMMENT -->
  <body>
  </body>
</html>
<hr>
```



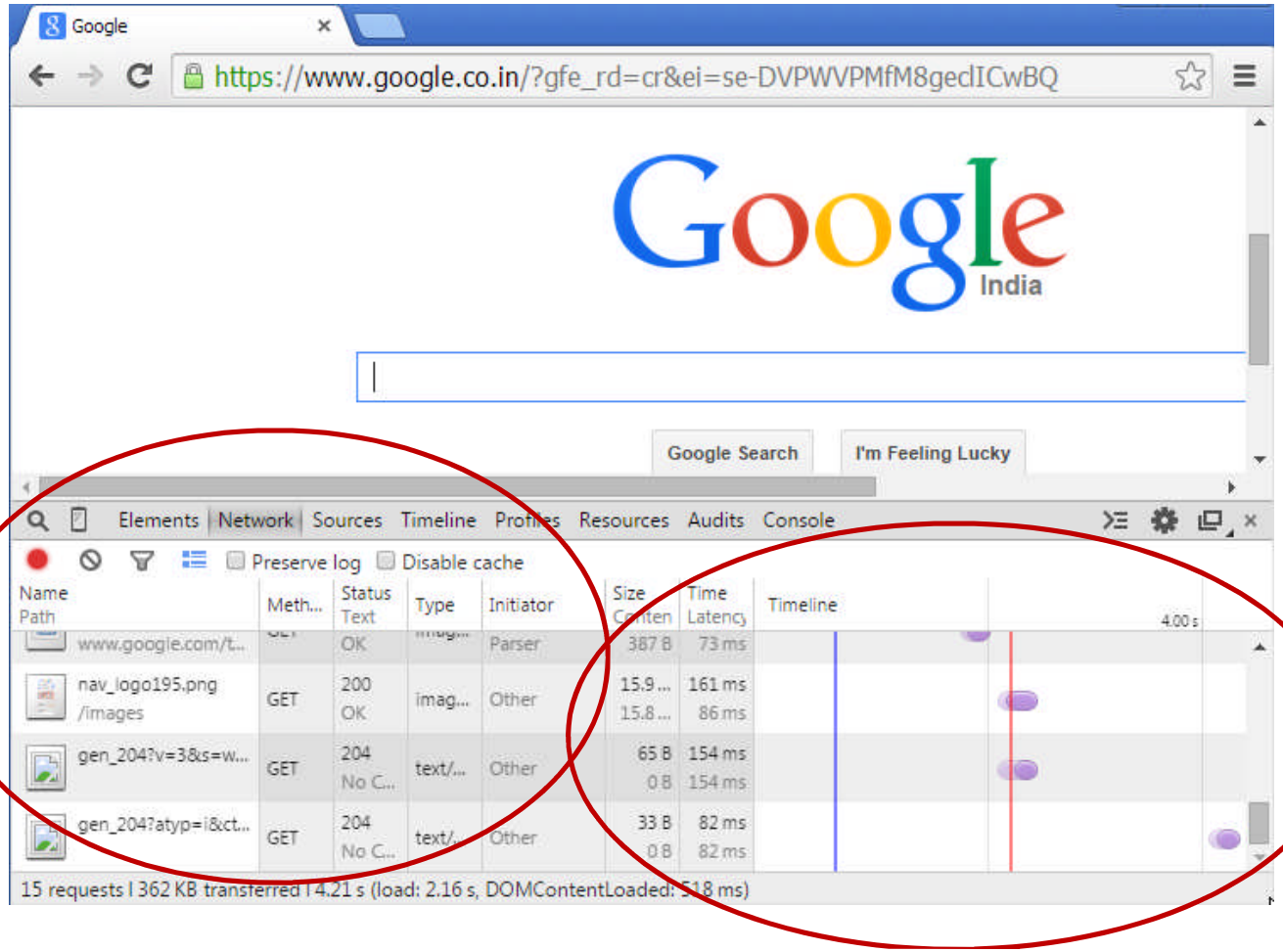
Inspecting HTML in Chrome
Right click > inspect element



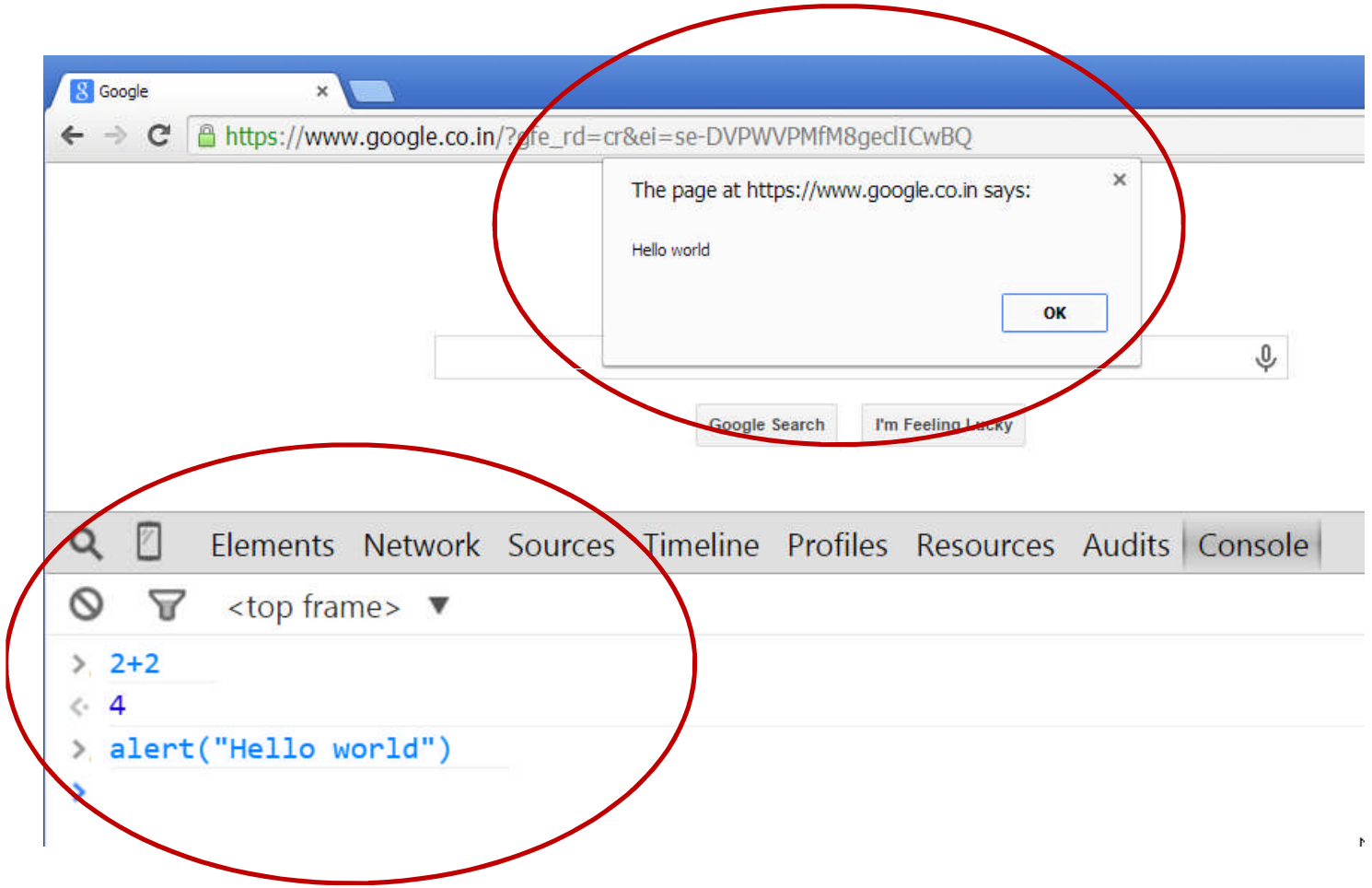
View HTML source and CSS



View network activity



Javascript console

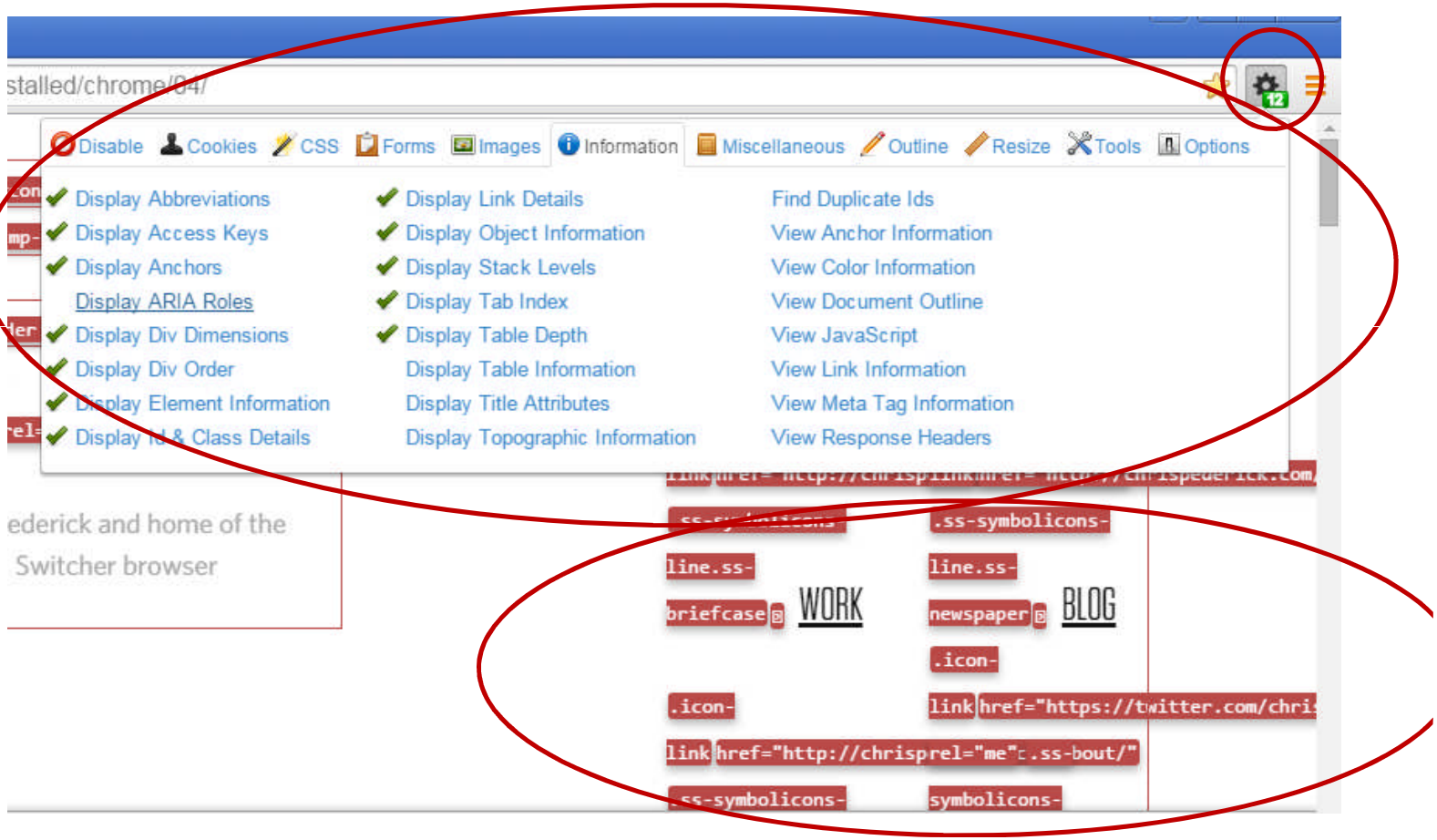


JavaScript

1. Google chrome developer tools
2. https://developer.chrome.com/devtools/docs/javascript-debugging (lots of info about chrome).

Chrome Developer Tools

(install from Google)



Git Exercises

mosh@hmi-tech.in
6/2017

Install git

On Windows install as admin:

1. c:/cygwin64 (cmdline git)
2. tortoise git (Shell integrated)
3. gitg (GUI)
4. gitk (GUI)
5. start run cmd [as admin]

Create a sandbox

```
c:\> bash
$ git config   --list   # see def options
$ git init --bare repo.git     # bare repo
# Setup two sandboxes for user a and b
$ git clone  repo.git usera
$ git clone  repo.git userb
$ ls
repo.git usera userb
```

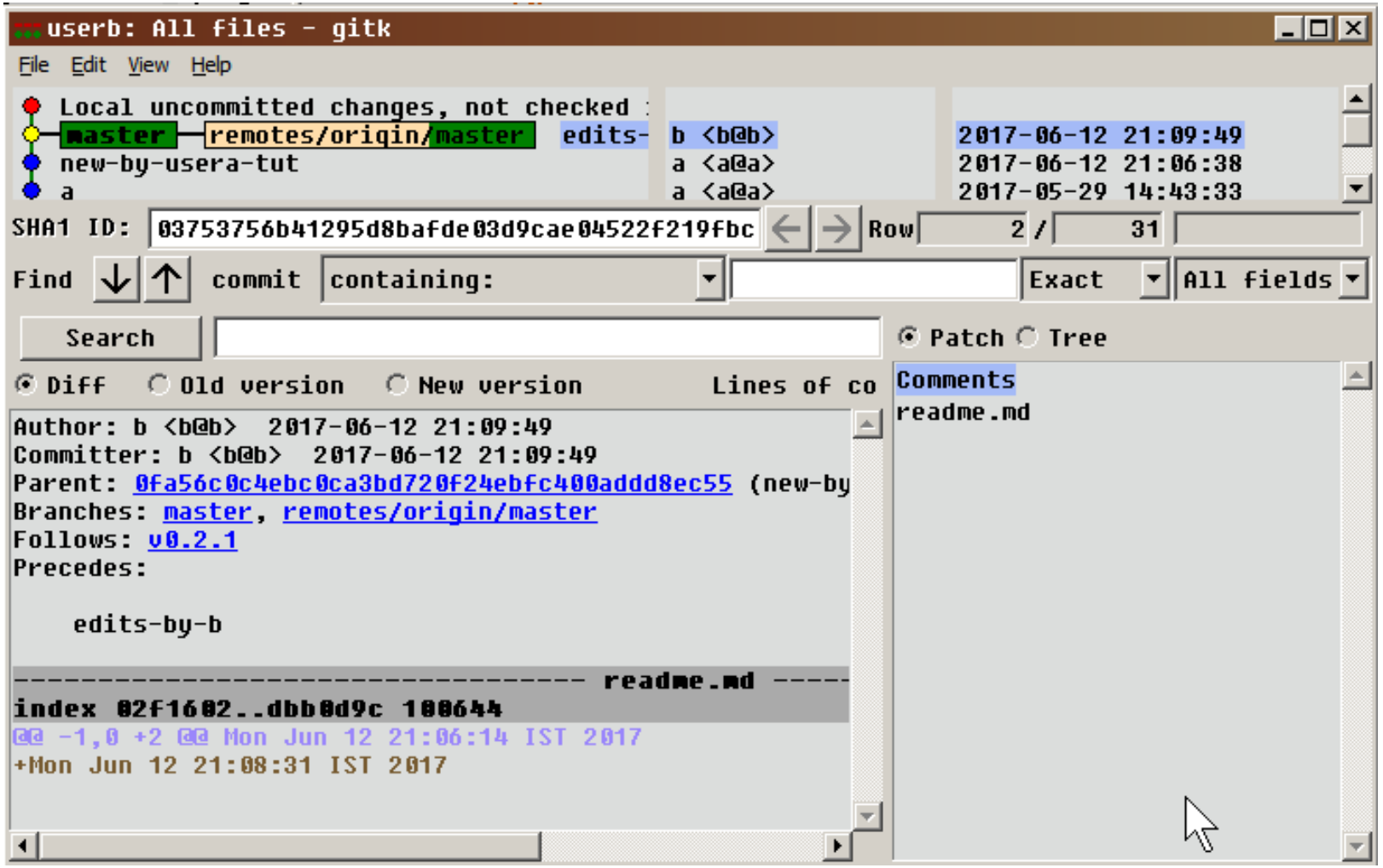
As user a

```
# user a adds a file
cd c:/src/tut/usera
git config user.name  a
git config user.email a@a.com
ls >  readme.md
git add readme.md
git commit -m new-by-usera readme.md
git push
```

As userb

```
cd c:/src/tut/userb
git config user.name b
git pull
ls  >> readme.md
git merge
git diff
vi readme.md # fix conflicts if any
git commit -m edits-by-b
git push
```

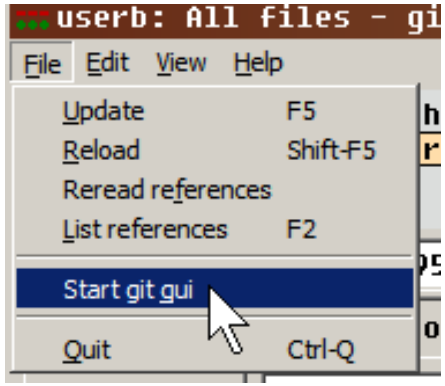
Start c:/tools/gitp/cmd/gitk.exe

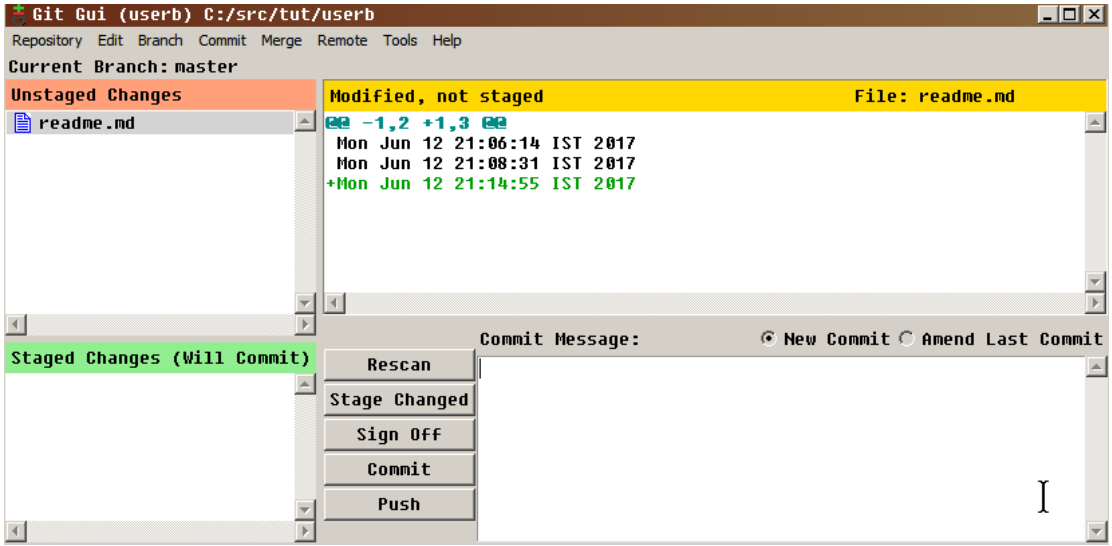


gitk > start git gui

$ cd c:/src/tut/userb

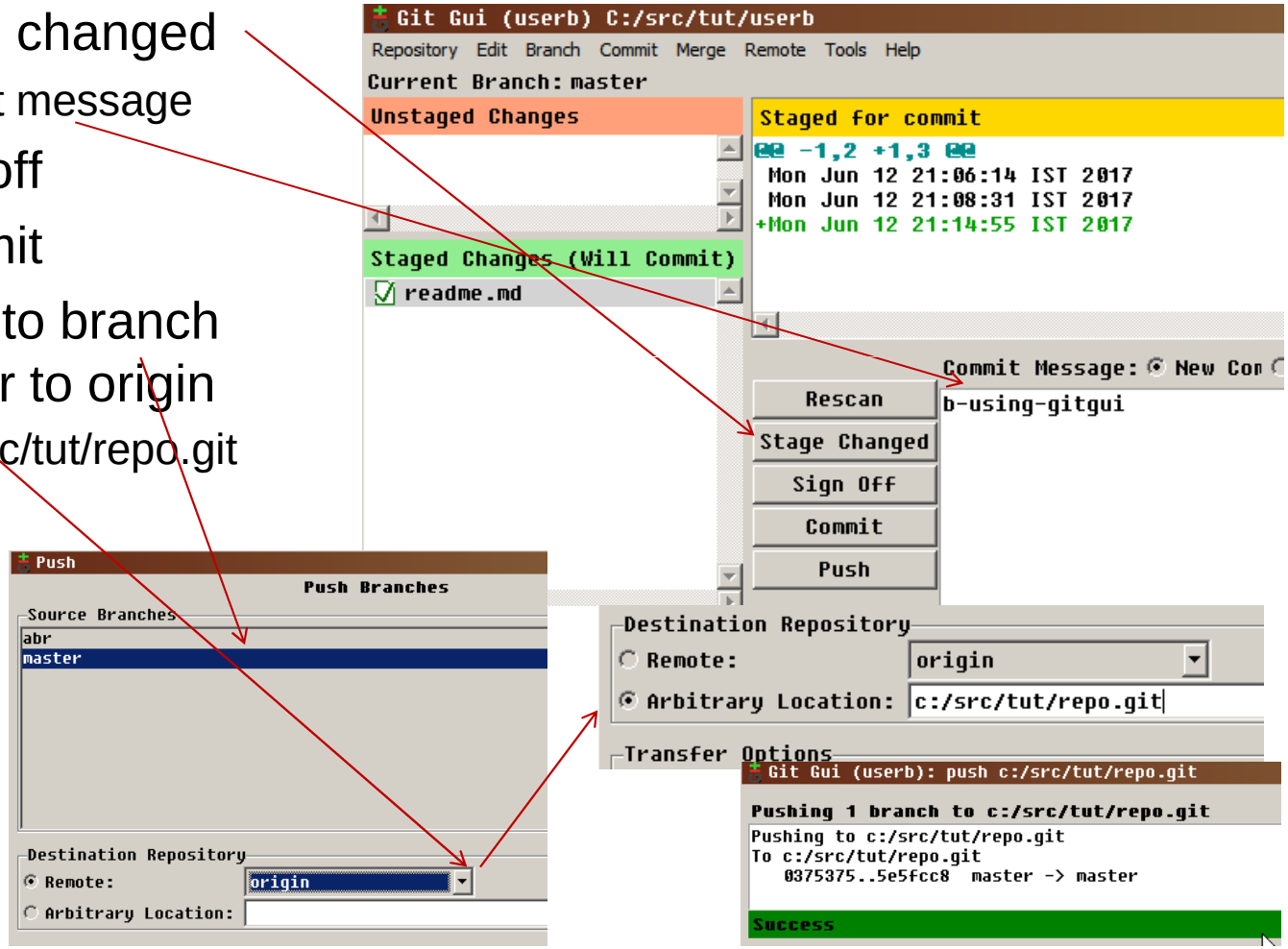
$ ls >> readme.md





Using git gui

1. stage changed
2. commit message
3. sign off
4. commit
5. push to branch
   master to origin
   or c:/src/tut/repo.git



tortoise git in windows explorer

$ cd c:/src/tut/userb

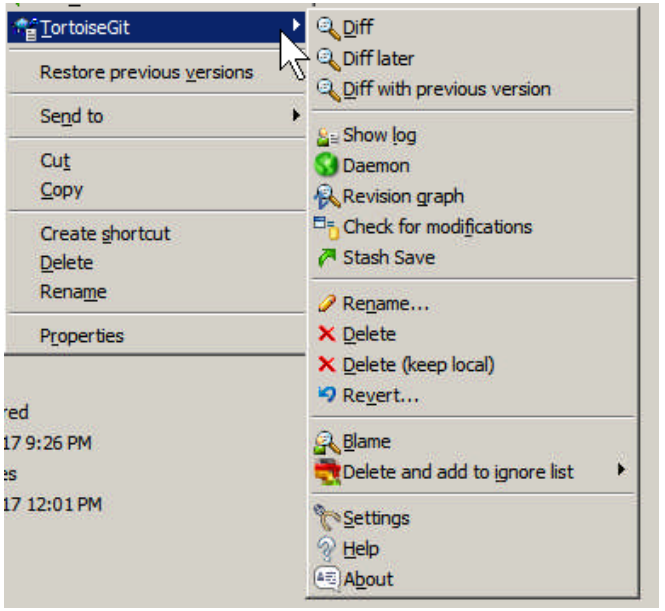
$ date >> readme.md

$ explorer .

[Right click on]
readme.md

[Commit]

Add message

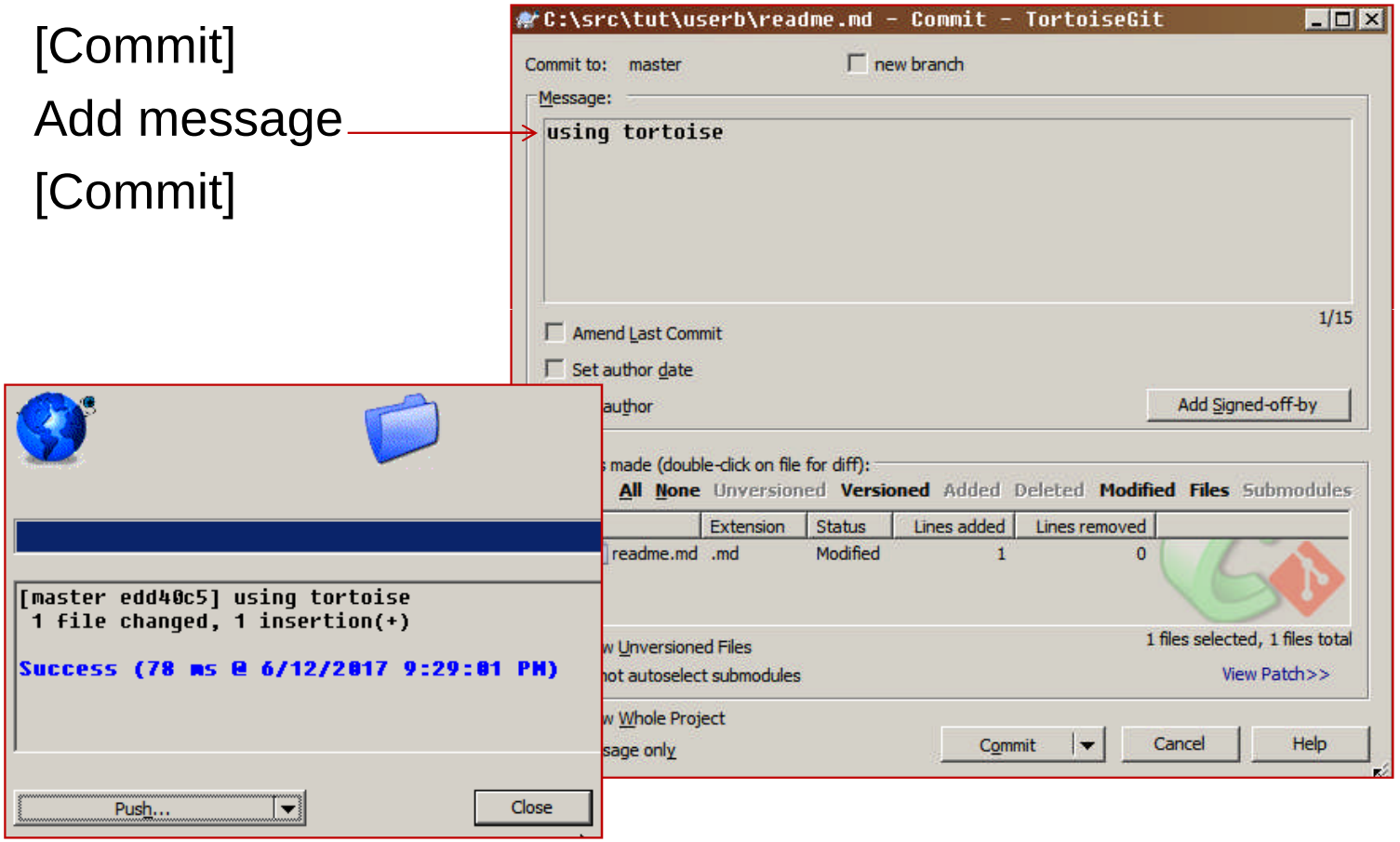


commit with tortoise git

[Commit]

Add message

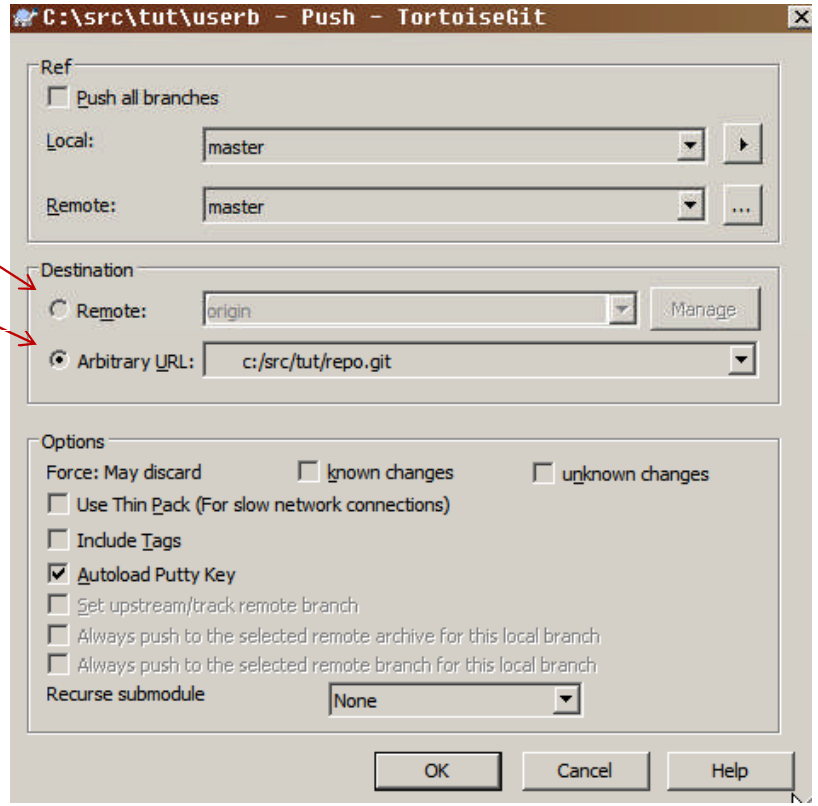
[Commit]



push with tortoise git

[Push] master to origin

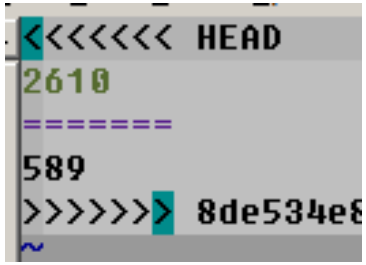
or to c:/src/tut/repo.git



Resolving conflicts

Both usera and userb modify the same line and try to checkin

```
$ cd c:/src/tut/usera
    $ echo $RANDOM readme.md
    $ git commit –m xx
    $ git push
$ cd c:/src/tut/userb
    $ echo $RANDOM > readme.md
    $ git commit –m yy
    $ git push   # ! [rejected]        master -> master (fetch first)
    $ git pull    # CONFLICT (content): Merge conflict
    $ edit readme.md   # delete <<==>> markers and fix.
    $ git mergetool
    $ git diff
    $ git add readme.md
    $ git commit –m merged
    $ git push
```



Branches

```
$ cd c:/src/tut/usera

$ git branch demo   # create a branch
$ git checkout demo # use it.
$ git branch –a # list all branches *demo..
$ date > readme.md   # make changes in the branch
$ git commit –m b1 readme.md
$ git push --set-upstream origin demo # push branch to repo

$ git checkout master   # use the main branch.
$ git merge demo  # merge demo into master.
$ git push origin master # push master branch to original repo.
```

References

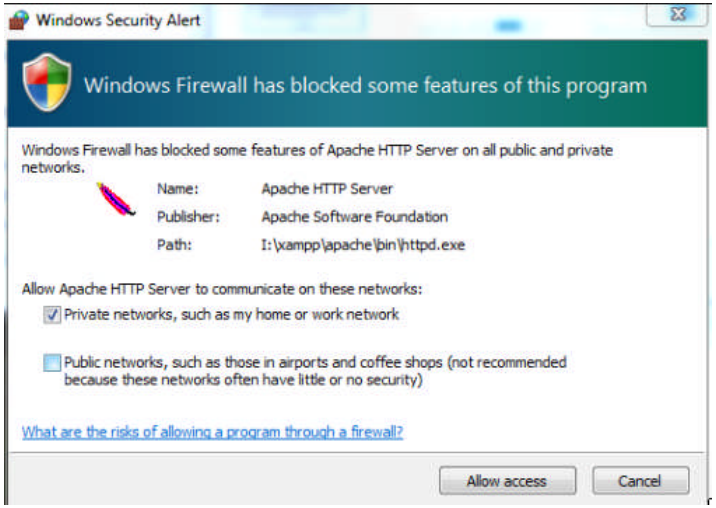
- google

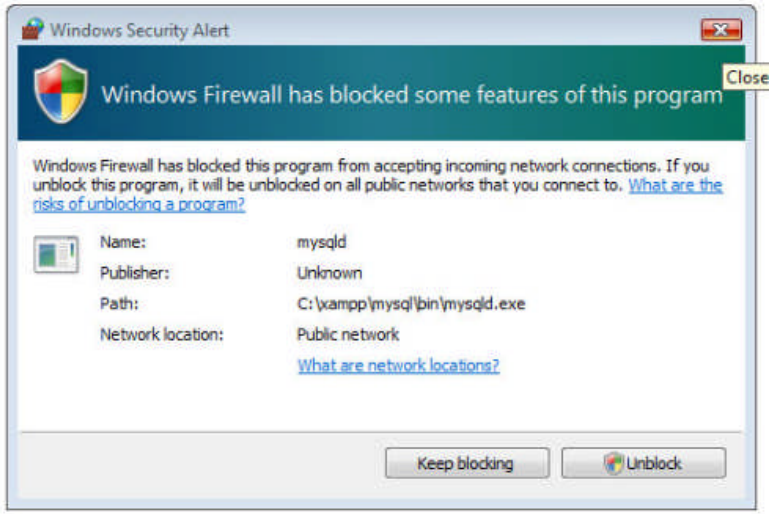
Xampp

XAMPP is the **acronym** for **X-cross platform**, **Apache**, **MySQL**, **PHP** and **Perl**

Download and Install Xampp

- From google, download and install xampp in c:/xampp
- Also look at wamp (web server)
- Firewall - block Apache and Mysql

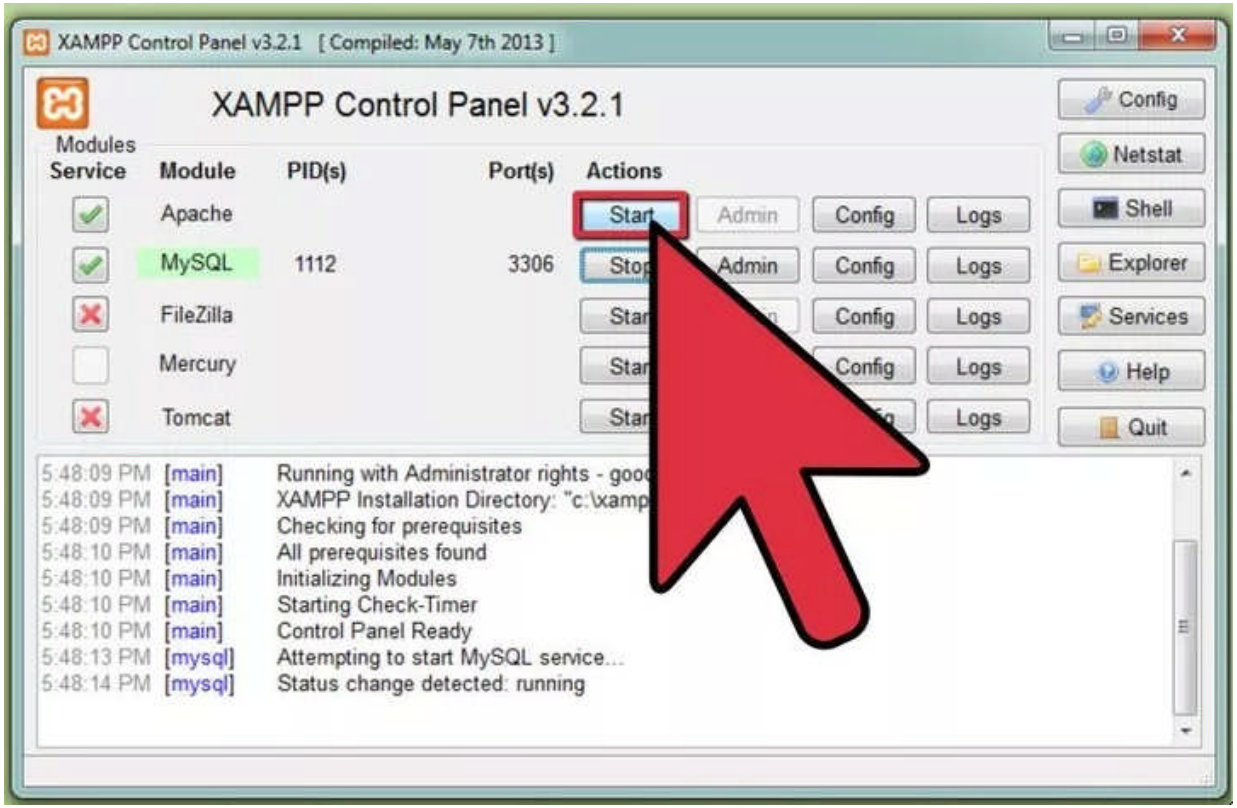




Start Xampp

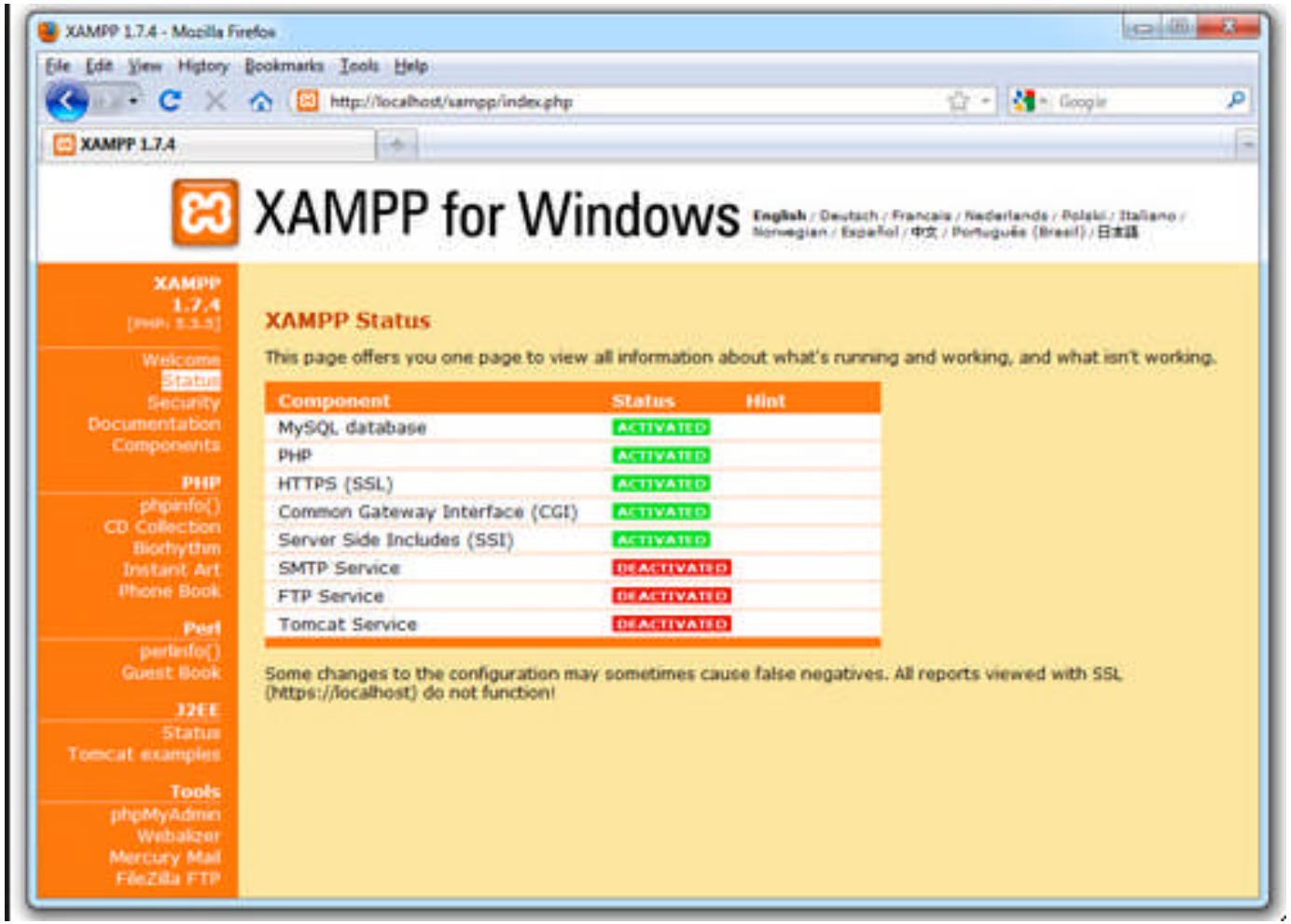
- Start Apache, MySql services.
- Check the website is working:
- chrome http://127.0.0.1
- The address of localhost is 127.0.0.1 is defined in C:/Windows/system32/drivers/etc/hosts

Start Apache and MySQL



See http://www.guru99.com/xampp-netbeans.html

http://localhost



Test webpage

Notepad++ > New file >
c:/xampp/htdocs/test.htm

```
<?php
 echo "Hello World!";
?>
```

> chrome http://127.0.0.1/test.htm

Put ip address in hosts file

> edit C:/Windows/system32/drivers/etc/hosts

  # Add ipaddress of your website in localhost

  127.0.0.1 your-website.you

> firefox http://your-website.you

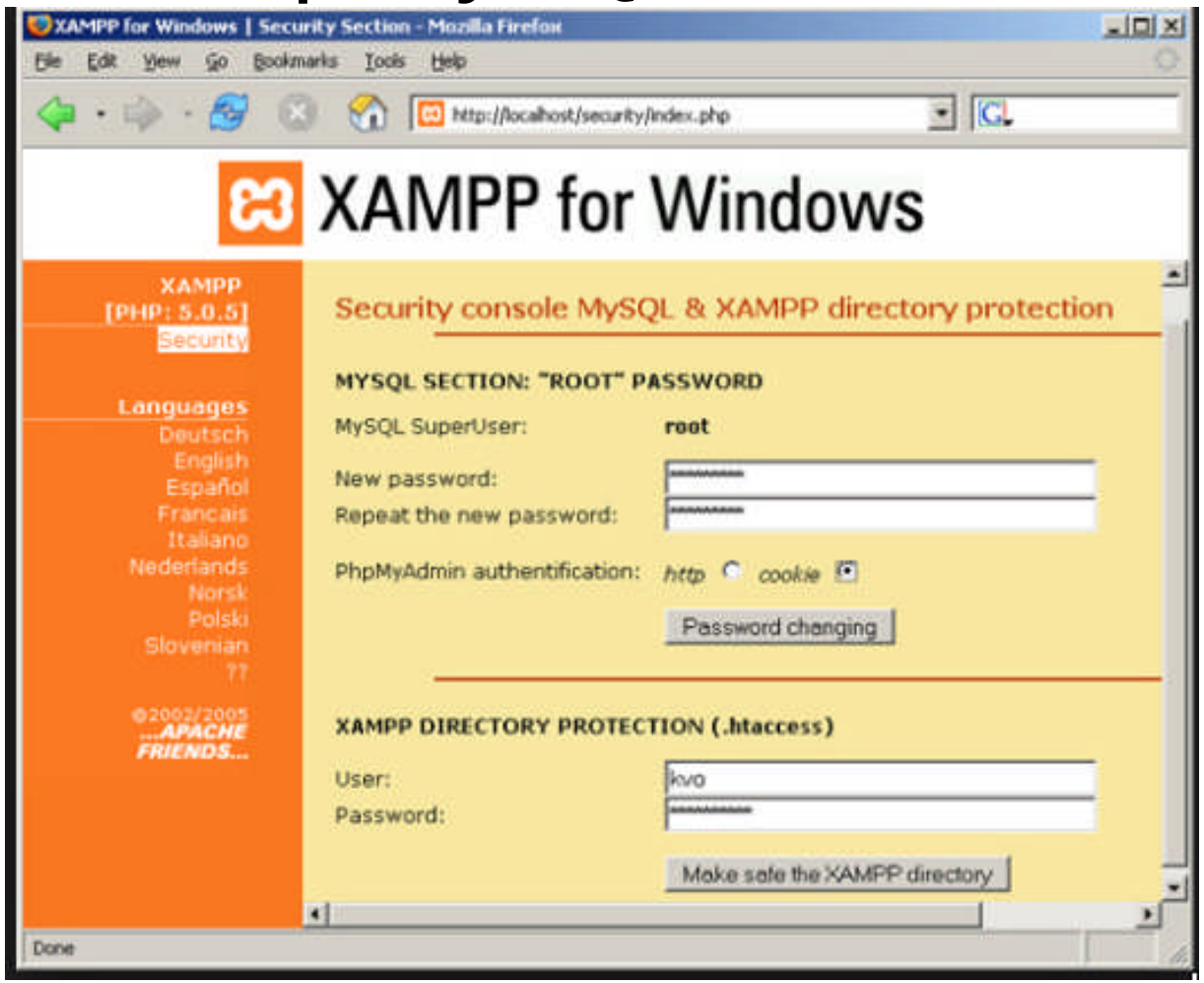
OR

> chrome http://127.0.0.1

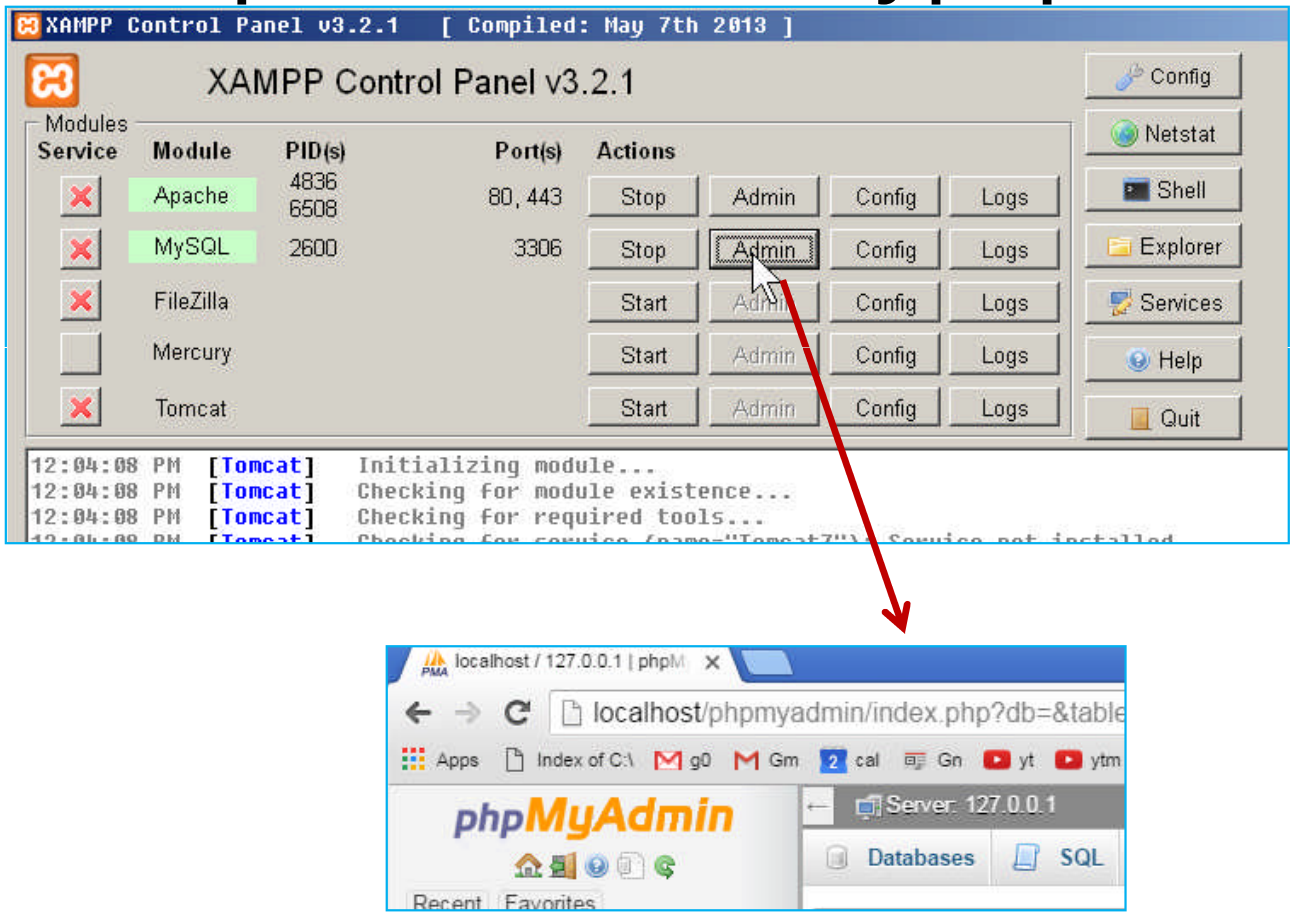
OR

> chrome localhost

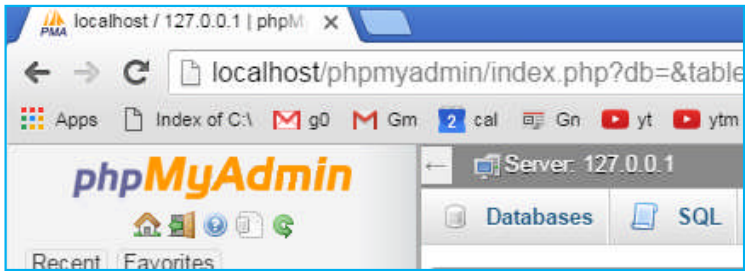
Setup MySQL account



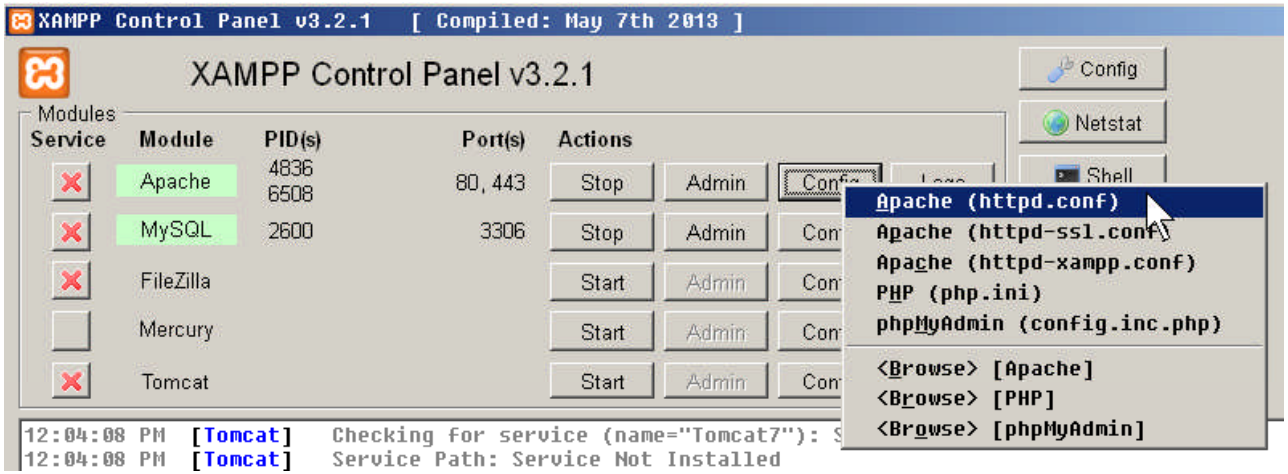
MySQL Admin > http://localhost/myphpadmin



http://localhost/myphpadmin



Export C:/ as http://localhost/C




```
# Add this text to httpd.conf
Alias /C "C:/"
    <Directory "c:/">
      Options Indexes FollowSymLinks Includes ExecCGI
      AllowOverride All
      Order allow,deny
      Allow from all
      Require all granted
    </Directory>
```

Chrome localhost/C .. To visit the C:/ via apache

Restart (stop/start) Apache

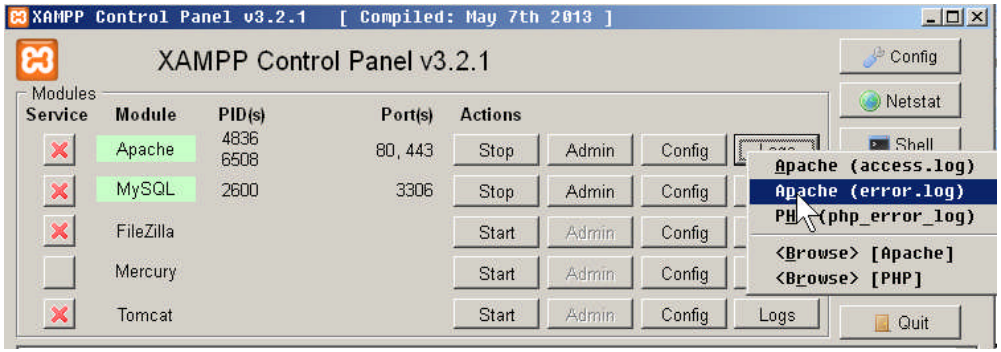
After any config changes in

c:/xampp/apache/conf/

Restart apache, check errors in

c:/xampp/apache/logs

Chrome http://localhost/C/



Xampp with SSL

Securing Apache with SSL
2011, 2017/3

Create server.key

C:\tmp> openssl genrsa -des3 -out server.key 1024

Generating RSA private key, 1024 bit long modulus

.....++++++

.....++++++

e is 65537 (0x10001)

Enter pass phrase for server.key: mosh

Verifying - Enter pass phrase for server.key: mosh

Generate server.csr

Type the command below on single line in cmd, use ^ to continue onto next line.

Note: If you are using wamp, your openssl.cnf maybe in different folder.

```
C:\tmp> openssl req  -new      ^
   -key server.key             ^
   -config C:\xampp\php\extras\openssl\openssl.cnf     ^
   -out server.csr
```

Enter pass phrase for server.key:mosh

Enter information … (next slide) your certificate request.

Enter Distinguished Name or a DN.

Generate server.csr

Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:MH
Locality Name (eg, city) [ ]:MUMBAI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:AZULARC
Organizational Unit Name (eg, section) [ ]:DEV
Common Name (eg, YOUR name) [ ]:MOSH
Email Address [ ]:mosh@hmi-tech.net
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password [ ]:boss
An optional company name [ ]:

Clear passwd from cert

```
c:\tmp> copy server.key server.key.org
c:\tmp> openssl rsa                  ^
        -in server.key.org           ^
        -out server.key
  Enter pass for server.key.org: mosh
```

Generating a Self-Signed Certificate

```
c:\tmp> openssl x509 -req       ^
    -days 365                   ^
    -in server.csr              ^
    -signkey server.key         ^
    -out server.crt
```

```
Signature ok
subject=/C=IN/ST=MH/L=MUMBAI/O=AZULARC/OU=DEV/CN=MOSH/emailA
    ddress=mosh@hmi-tech.net
Getting Private key
```

Installing the Private Key and Certificate

> copy server.crt c:\xampp\apache\conf\

> copy server.key c:\xampp\apache\conf\

Edit with

notepad c:/xampp/apache/conf/extra/httpd-ssl.conf

add these two lines:

SSLCertificateFile "conf/server.crt"

SSLCertificateKeyFile "conf/server.key"

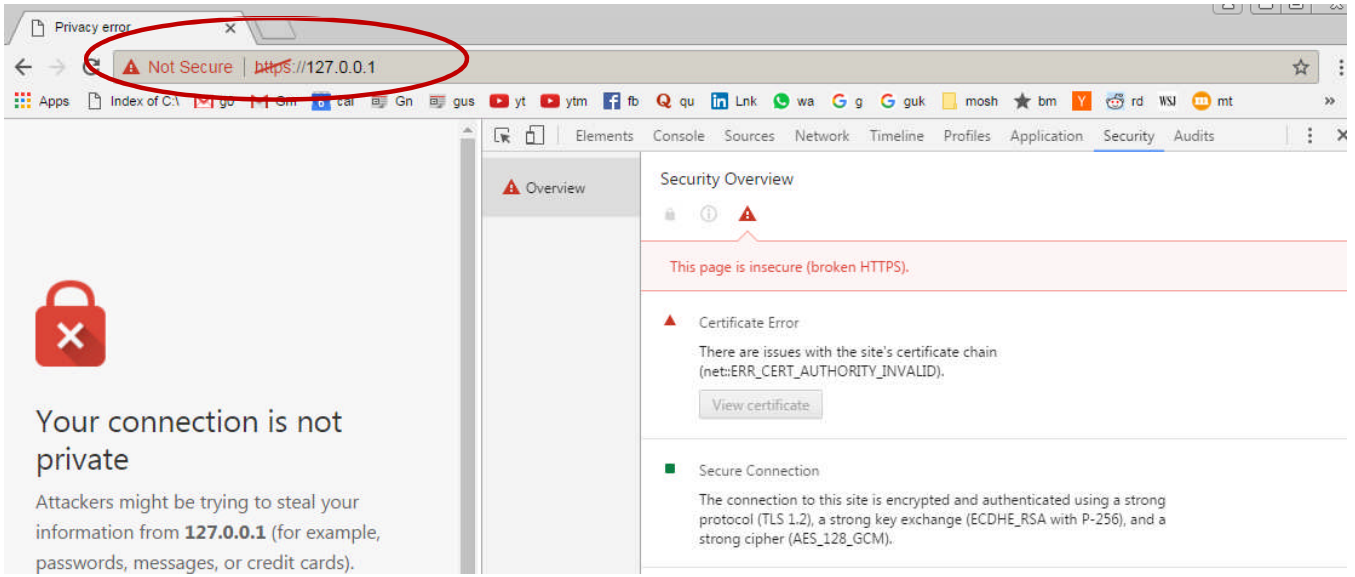
Restart Apache and Test

Start> chrome

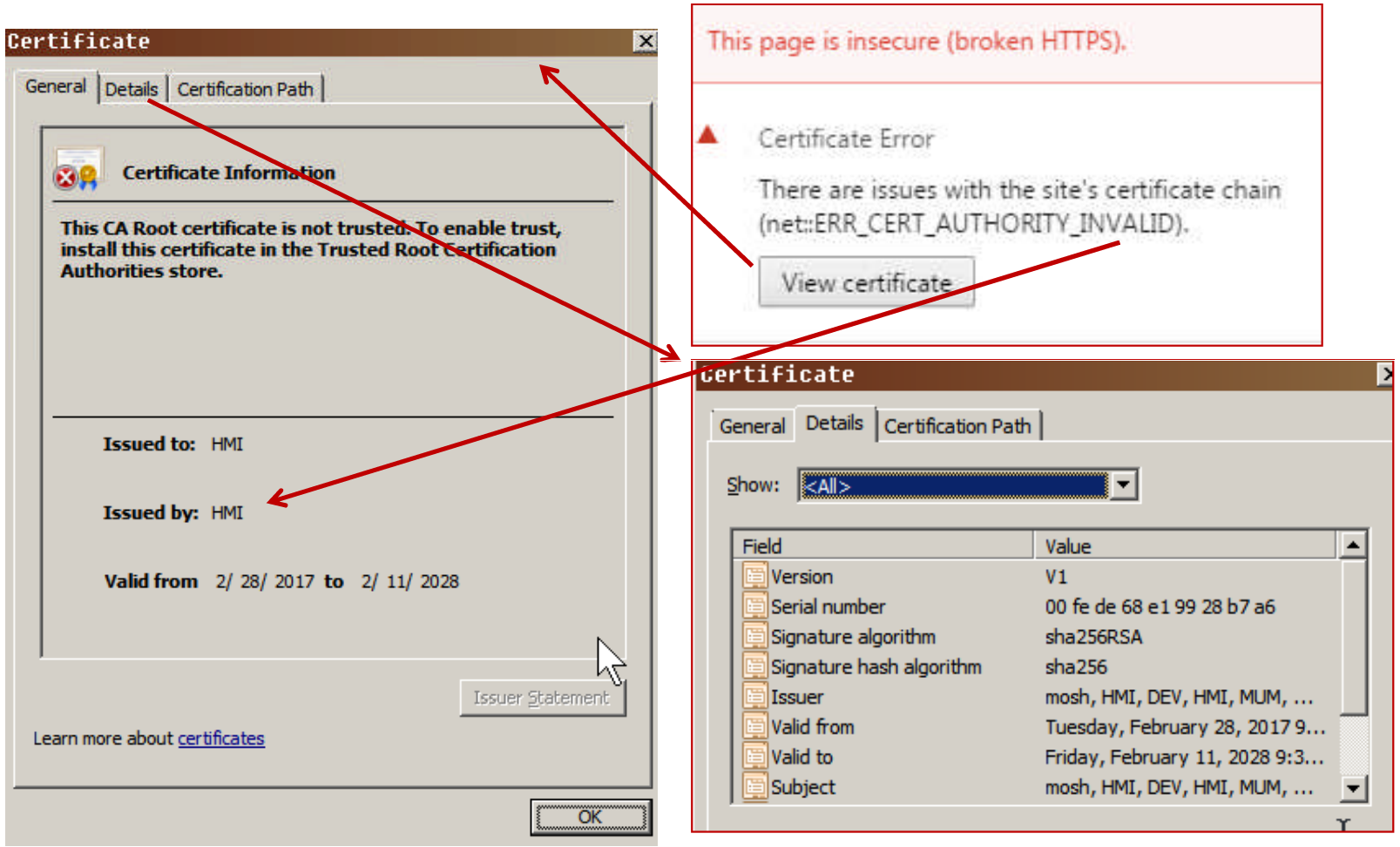
> https://localhost/xampp/

> https://127.0.0.1

Right click and inspect the webpage and cert



View Cert from chrome



Unit Testing
php and github

mosh@hmi-tech.in
6/2017

Setup on windows

1. Install c:\xampp
2. Install c:\cygwin64 (git, php,bash).
3. Install puttygen
   https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Start cmd

> start > run > cmd [run as admin]

> mkdir c:\github

> cd c:\github

> git clone https://github.com/moshahmed/php

> cd c:\github\php

Run the tests

> cd c:\github\php\unittest

> make

C:/xampp/php/php.exe phpunit.phar CalculatorTest.php

…etc…

Create your ssh private keys

C:\> bash

1. bash$ ssh-keygen

   Create your private key in ~/.ssh/id_rsa

   and public key in ~/.ssh/id_rsa.pub

2. Note down your passphrase.

3. Tell ssh your server and keys:

bash$ cat ~/.ssh/config

```
Host github.com
    IdentityFile ~/.ssh/id_rsa
```

Setup your keys for git

Tell git about your key via env variable

```
c:\>  setx GIT_SSH_COMMAND
      "ssh -F ~/.ssh/config -i ~/.ssh/id_rsa" -m
```

Using github

1. Create your github.com account
2. Right profile icon > settings > ssh > new ssh key and
3. Paste ~/.ssh/id_rsa.pub into github
4. Test your github connection

```
bash$ ssh -F ~/.ssh/config -i ~/.ssh/id_rsa git@github.com
passphrase for id_rsa:***
Success
```

Save in Repo

- Use browser to create a github repo called 'php'

> cd c:\github\php

> git remote set-url origin git+ssh://git@github.com/moshahmed/php.git

> git add ● # dot is pwd.

> git commit –m "tutorial"

> git push

Exercise

- Fix all the unittests.

> make

  [till no assertions fail]

> git diff

> git add .

> git commit –m "fixing tests"

> git push

References

- https://github.com/moshahmed/php

Software Engineering

mosh@hmi-tech.in
6/2017

Paradigms

- Write specs and docs
- TOP DOWN and BOTTOM UP Development
- Pair programming.

Agile programming

Top down design

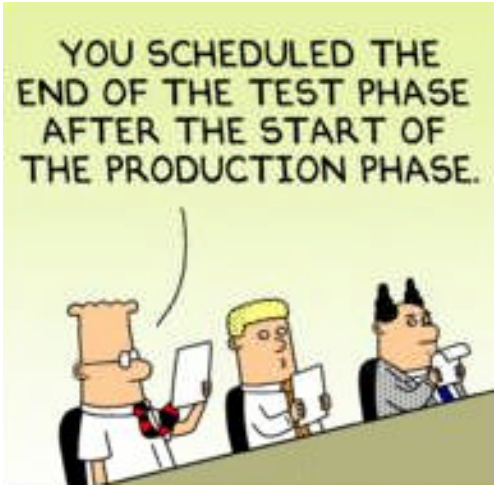
- Break it into functions
- Create dummy functions with in/out
- Write assertions
- Fill in the boxes top down and bottom up.

Testing

- Ad hoc
- Manual GUI testing.
- Automated testing.
- Developer – unittest.
- Integration tests.
- A/B Testing

Testing

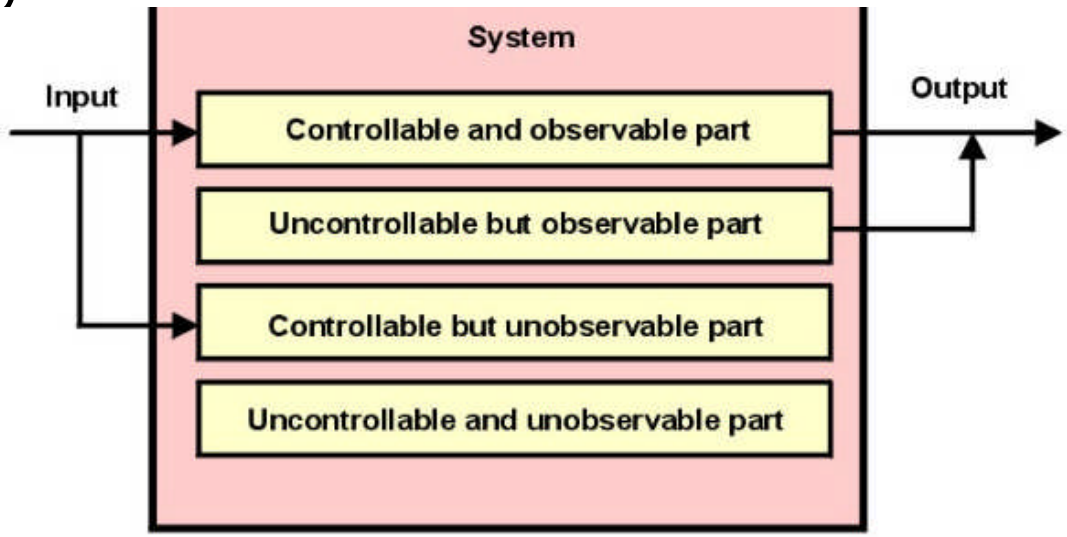

I DID A–B TESTING AND FOUND THE SEARCH TERMS THAT BRING THE MOST PEOPLE TO OUR SITE.
A–B


YOU SCHEDULED THE END OF THE TEST PHASE AFTER THE START OF THE PRODUCTION PHASE.


I SPENT THE WEEK WRITING A TEST SCRIPT FOR OUR PRODUCT.
AND I WROTE A TEST SCRIPT TO TEST DILBERT'S TEST SCRIPT.
DilbertCartoonist@gmail.com

TDD

- Test Driven Development: Write tests before writing code.
- Design for Testability
  - Observability
  - Controlability
- Examples?



Test cases

- Write test cases for each module/box
  - usual case for debugging during development
  - Corner cases, e.g. 1*0, 1*-1,
  - Expected failures, e.g. 1/0
- Run php unittest as you/others add features, upgrade OS.
- Save source, tests, test results in daily in a db to track progress.

Comments are not code

- Turn comments into assertions
- Turn comments into function names.
- What are mocks, Fixtures?

Mocks and Fixtures

- **Mock** is a method/object that simulates the behavior of a real method/object in controlled ways. **Mock** objects are used in unit testing. Often a method under a test calls other external services or methods within it. These are called dependencies.
- A **test fixture** is a fixed state of a set of objects used as a baseline for running **tests**. The purpose of a **test fixture** is to ensure that there is a well known and fixed environment in which **tests** are run so that results are repeatable. ... Preparation of input data and setup/creation of fake or mock objects.

Assertions in php

```
// Active assert and make it quiet
assert_options(ASSERT_ACTIVE, 1);
assert_options(ASSERT_WARNING, 0);
assert_options(ASSERT_QUIET_EVAL, 1);

assert( '$a > 0');
```

Assert Options

Option	INI Setting	Default value	Description
ASSERT_ACTIVE	assert.active	1	enable assert() evaluation
ASSERT_WARNING	assert.warning	1	issue a PHP warning for each failed assertion
ASSERT_BAIL	assert.bail	0	terminate execution on failed assertions
ASSERT_QUIET_EVAL	assert.quiet_eval	0	disable error_reporting during assertion expression evaluation
ASSERT_CALLBACK	assert.callback	(NULL)	Callback to call on failed assertions

Building (compile and link)

- Developer builds
- Fix all warning (-Wall, /W4)
- Build with multiple compilers.
- Continuous build.
- Build verification tests

Source control

- Use Git / Perforce / SVM / CVS
- Track changes and history
- User Permissions (read / write)
- Offsite backups (daily/weekly/monthly)
- Code reviews
- Style guide.

Issue tracking

- Bug database.
- Assign to dev / tester with priority.
- testcases for debugging and later re-use.

Performance

- Performance test
- Micro-benchmarks
- Benchmarking
  - Average case
  - Worse case
  - Best case
  - Load test
- Measure time, memory, disk, network
- Compare and save results in db.

Servers

- Dev
- Canary
- Production
- Regional replication

Test to deployment

- Logging
- Breakpoints
- Assertions
- Unittest
- Integration test
- Automated test
- Build verification test
- Randomized test
- Load test
- Canary test
- Development server
- Production server
- Interactive GUI test
- Monitoring webservice

Monitoring

- Logging
- Sampling servers
- Time series
- Moving Averages
- Aggregating series
- Charting
- Alerts
- Paging

Mythical Man month

- Adding more people to late projects delays it more.
- Timeline is not linear
- Wo/Man-power is not linear.
- Code value != number of lines

References

1. Github google/googletest
2. Github google/benchmark
3. Apache bench (ab)

   https://www.sitepoint.com/stress-test-php-app-apachebench/

TDD in php for 1000!

Test Driven Development

mosh@hmi-tech.in

Azularc, 6/2017

Problem for TDD

- Design a php class to computer 1000! using basic operations in php only (no libraries).

Problem statement

- User input: N
- Display: N! (1*2*3*..*N)

TDD

- Write Specs.
- Test Driven Development: Write tests before writing code.
- Design for Testability
  - Observability
  - Controlability

Testing

- Ad hoc
- Manual GUI testing.
- Automated testing.
- Developer
  - Assertions
  - Assumptions.
  - unittest.
- Integration tests.

Exercise 1

- Write php code to multiple two numbers input by user in the web browser.

- Write unittest for it.

Exercise 2.

- Write a php program using just strings and basic operators to compute 100!
  100!=(100*99*98*…*2*1)

- Estimate size of output?
- Write a unittest for the code.
  - Enumerate the corner cases.

Top down design

- How to compute N!
- Design the functions
- Write few testcases for each function.

Multiplying large numbers In php

A

B

multiply

A=111111111111111111111111
B=222222222222222222222222
A+B =2.33333333333E+24
big Sum =233333333333333333333333
A*B =2.46913580247E+47
big Mult =246913580246913580246913308641975308641975308642

Form

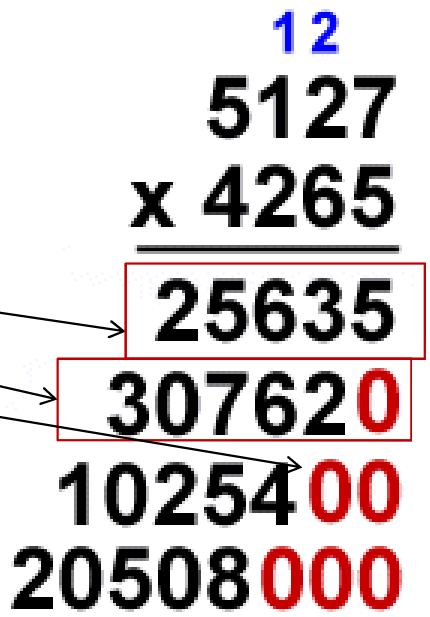
```
1.  <form method="GET">
2.    A <input type="text" name="A" size=100
3.      value="<?php echo isset($_GET['A']) ? $_GET['A'] : '1' ?>"
4.      > <br>
5.    B <input type="text" name="B" size=100
6.      value="<?php echo isset($_GET['B']) ? $_GET['B'] : '2' ?>"
7.      > <br>
8.    <input type="submit" value="multiply" >
9.   </form>
```

Show results

```
1.  <?php
2.      $A = $HTTP_GET_VARS["A"];
3.      $B = $HTTP_GET_VARS["B"];
4.      show('A',$A);
5.      show('B',$B);
6.      show('A+B ', $A + $B);
7.      show('big Sum ', sum($A,$B) );
8.      show('A*B ', $A * $B);
9.      show('big Mult ', mult($A,$B) );
10.  ?>
```

Mult two big numbers

```
1.   /* mult_str("12","34")  is "408" */
2.    function mult_str($A,$B) {
3.      $M = '0'; $b = 0;
4.      $shifter = '';
5.      while( last_digit($B,$b) ) {
6.        $d = mult_one_digit($A, $b);
7.        $M = sum_str( $d. $shifter , $M);
8.        $shifter = $shifter.'0';
9.      }
10.     return $M;
11.   }
```



Example

Sum two big numbers, with carry

```
1.  /* sum_str("11", "22") is "33" */
2.   function sum_str($A, $B) {
3.     $a = 0; $b = 0; $carry = 0; $S = '';
4.     while( strlen($A)>0 || strlen($B)>0) {
5.       last_digit($A, $a);
6.       last_digit($B, $b);
7.       $t = $a + $b + $carry;
8.       $carry = (int) ($t / 10);
9.       $t = $t % 10;
10.       $S = $t . $S;
11.    }
12.    if( $carry > 0)
13.      $S = $carry . $S;
14.    return $S;
15.  }
```

Multiply one digit at a time

```
1.  /* mult_one_digit("12",2)  is "24" */
2.  function mult_one_digit($A, $b) {
3.    $S = '0';
4.    while( $b--  > 0) {  // multiply by repeated addition
5.      $S = sum_str($S,$A);
6.    }
7.    return $S;
8.   }
```

Get least significant digit

```
1.  // last_digit("1234",...) .. returns true
2.  //       and (A="123",a=4) (call by reference).
3.    function last_digit(&$A, &$a) {
4.      $a = 0;
5.      if (strlen($A)<1)
6.        return false;
7.      $a = (int) substr($A,-1);
8.      $A = substr($A,0,-1);
9.      return true;
10.   }
```

BigMultClass

```
1. class BigMultClass {
2.     function mult_str($A,$B) {
3.     // printf("\n%s::%s:%s:%d\n", __CLASS__,
4.     //  __FUNCTION__, __FILE__, __LINE__);
5.     $M = '0'; $b = 0;  $shifter = '';
6.     while( $this->last_digit($B, $b) ) {
7.         $d = $this->mult_one_digit($A, $b);
8.         $M = $this->sum_str( $d. $shifter, $M);
9.         $shifter = $shifter.'0';
10.    }
11.    return $M;
12.  }   // etc
13.}
```

Unittesting: BigMultTests

```
<?php
require 'BigMultClass.php';
use PHPUnit\Framework\TestCase;
class BigMultTests extends TestCase {
    private $bmc;
    protected function setUp() { $this->bmc = new BigMultClass(); }
    protected function tearDown() { $this->bmc = NULL; }
    public function testMult() {
            $this->assertEquals( "408", $this->bmc->mult_str("12","34") );
   }
}
```

```
# https://github.com/moshahmed/php/tree/master/unittest
c:\> C:/xampp/php/php.exe  phpunit.phar  BigMultTests.php
PHPUnit 6.1.2 …
OK (1 test, 1 assertions)
```

MySQL

mosh@hmi-tech.in
6/2017

MySQL security
sql over ssh tunnel

1. Make a ssh tunnel from forward local port 3333 to

   remote server port 3306

   where MySQLd is listening for local connections.

C:\> ssh -F ~/.ssh/config -i myssh.key -L 3333:localhost:3306 user1@remote.com

C:\> netstat -a | grep 3333   # TCP localhost:3333 listening

2. Use the tunnel from localhost to remote host.

C:\> mysql -h localhost --port=3333 -u use1 -p"pass"

mysql> show databases;

…

Secure backup of db

```
$ cat dump.cnf
   [mysqldump]
    host=localhost
    user=root
    password="xyz"
$ chmod og-rwx dump.cnf # keep mysql password
  private.
$ mysqldump --defaults-extra-file=dump.cnf --all-
  databases | gpg --encrypt-r userid –output
  dump.sql.gpg
1. never use -p password .. visible to all with
  ps command.
2. Separate backup of sensitive info.
```

Db security

- Keep sensitive information in separate tables, indexed by common keys.
- Encrypt and store sensitive information like credit-card in db.
- Hash (sha256) and Store only passwords in db.
- Keep transaction logs, login logs, backup logs offsite.
- Keep a separate db for billing info, with foreign userid-key in main db.

Privacy

- User userids (sha256(emailid)) in db, instead of emailids (sensitive information).
- Ip-address is private info, encrypt it.
- Consult lawyer of each country your website is used in, for privacy laws.
- Privacy policy: list how long you keep cookies and ip-address logs.

MyISAM vs InnoDb

- My-ISAM (old).
  1. Supports table level locking
  2. Faster than InnoDB
  3. Does not support foreign keys
  4. Stores table, data and index in different files
  5. Does not support transactions (no commit with rollback)
  6. Useful for more selects with fewer updates
- Inno-DB (new).
  1. Support row level locking
  2. slower than MyISAM
  3. Supports foreign keys
  4. Stores table and index in table space
  5. Supports transactions

MySQL Performance

> show full processlist;

> explain select * from wp_posts;

> SHOW VARIABLES LIKE 'have_query_cache';

> SHOW STATUS LIKE 'Qcache%';

In */etc/mysql/my.cnf enable logging of*

- #log_slow_queries = /var/log/mysql/mysql-slow.log
- #long_query_time = 2
- #log-queries-not-using-indexes

SQL Injection

- See http://www.unixwiz.net/techtips/sql-injection.html

References

1.https://syedali.net/engineer-interview-questions/ .. SQL Perf

2.http://www.unixwiz.net/techtips/sql-injection.html .. SQL Injection

Node js

Node.js

1. Created by Ryan Dahl in 2009
2. Development && maintenance sponsored by Joyent
3. Licence MIT
4. Version 6.11 (2017-6)
5. Based on Google Chrome V8 Engine
6. Data is generated on the server, transferred to the client and displayed by the browser
7. ServerSide: nodejs, PHP, JSP, ASP.net, Rails, Django, java servelets.

Advantages

1. Non Blocking I/O
2. V8 Javascript Engine
3. Single Thread with Event Loop
4. Millions of modules
5. Windows, Linux, Mac
6. Same Language for Frontend and Backend
7. Active community

What is node.js?

- Asynchronous I/O framework
- Core in C++ on top of v8; remaining in js.
- 1000s of concurrent connections with minimal overhead (cpu/memory) on a single process
- NOT a web framework,
- NOT a language.

Getting Started….. Hello World

• Download and install nodejs in c:\tools\nodejs

Install modules/library

$ npm install *modulename* # local to app

$ npm install cheerio –g # for global

$ cat hello.js

```
  console.log("Hello World");
```

$ c:/tools/nodejs/node.exe hello.js

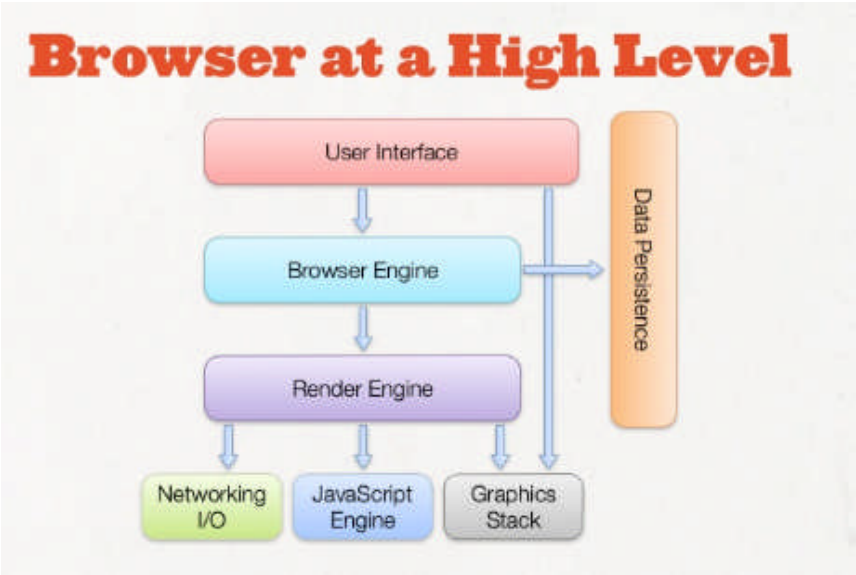
Hello World

JavaScript Engines……

Every browser has **its own VM**

Firefox? **Spidermonkey**
Internet Explorer? **Chakra**
Chrome? **V8**
Safari? **JavaScriptCore**
Opera? **Carakan**

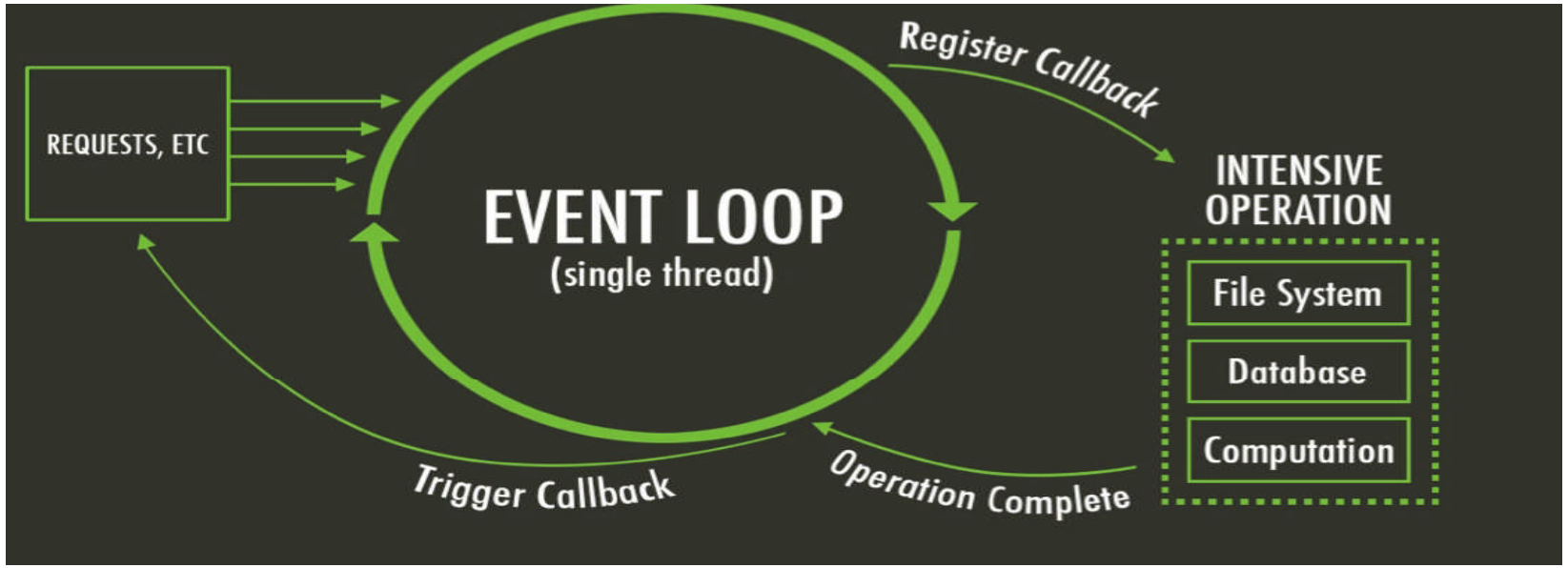
Also **Rhino**, stand alone



The ideas

1. Single thread async-processing, instead of classical multithreading. Minimize overhead & latency, maximize scalability.
2. Scale horizontally instead of vertically
3. Ideal for applications that serve a lot of requests but don't use/need lots of computational power per request.
4. Not for heavy calculations / massive parallel processing.
5. Less problems with concurrency

Node.js Event Loop

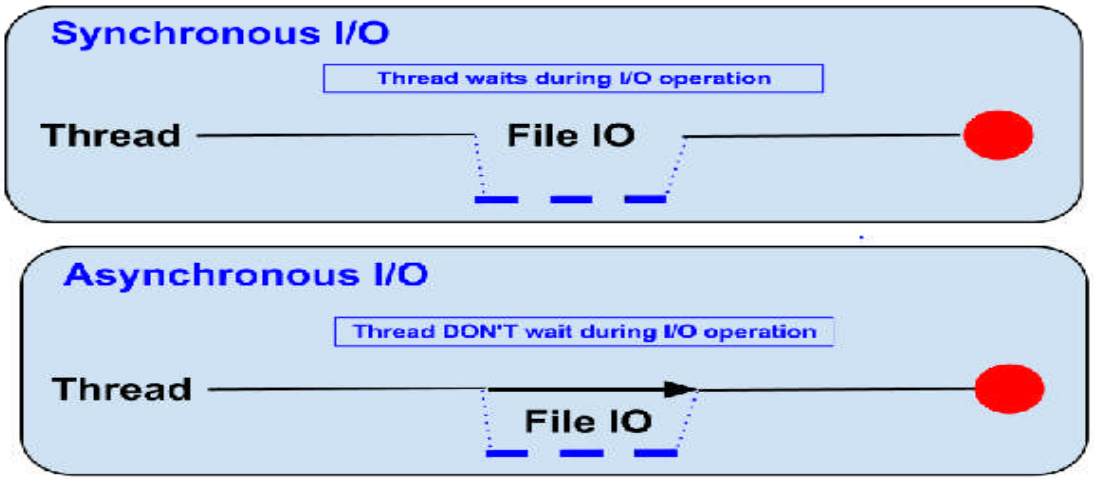


There are a couple of implications of this apparently very simple and basic model

- Avoid synchronous code at all costs because it blocks the event loop
- Which means: callbacks, callbacks, and more callbacks

Blocking vs Non-Blocking……

Example: Read data from file and show data



Blocking IO file reader

```
// Read data from file and print on screen
var data = fs.readFileSync( “test.txt” );
// blocks until file is read.
console.log( data );
```

Non-Blocking file reader with Callback

```
// Continue doing other things
// while waiting for data to be read.
fs.readFile(    “filename.txt”,
/* anonymous callback */ function( err, data ) {
console.log(data);  }
);
```

When to use it ?

- Chat/Messaging
- Real-time Applications
- Intelligent Proxies
- High Concurrency Applications
- Communication Hubs
- Coordinators

Using a module in nodejs

1. var http = require(‘http’);
2. var fs = require(‘fs’);
3. var express = require(‘express’);

Debugging

```
c:\> node debug program.js
```

Example

```
C:\mosh\web\nodejs$ node debug server-web.js
< Debugger listening on [::]:5858
connecting to 127.0.0.1:5858 ... ok
break in C:\mosh\web\nodejs\server-web.js:4
> 4 var http = require('http');
  5 var fs = require('fs');
  6 var url = require('url');
debug> help
Commands: run (r), cont (c), next (n), step (s), out (o), backtrace (bt), setBreakpoint
(sb), clearBreakpoint (cb), watch, unwatch, watchers, repl, exec, restart, kill, list,
scripts, breakOnException, breakpoints, version
debug> r
```

…

References

- Introduction https://www.tutorialspoint.com/nodejs/nodejs_quick_guide.htm
- Debugging https://www.javatpoint.com/nodejs-debugger