

Matrix computation in Excel

Matrix Commands in Excel

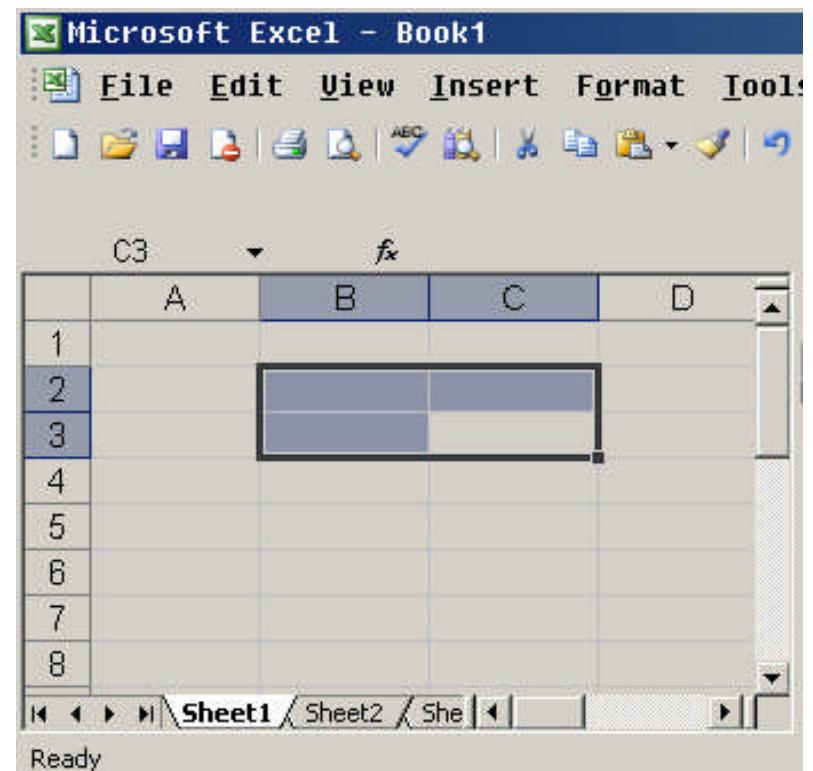
Excel can perform some useful matrix operations:

- Addition & subtraction
- Scalar multiplication & division
- Transpose (**TRANSPOSE**)
- Matrix multiplication (**MMULT**)
- Matrix inverse (**MINVERSE**)
- Determinant of matrix (**MDETERM**)

As well as combinations of these operations.

Cells

- Most Excel formula require you to name one or more cell ranges e.g. B2:C3.
- You can type these in directly or select them using the mouse.
- However, it is often better to use a named range.

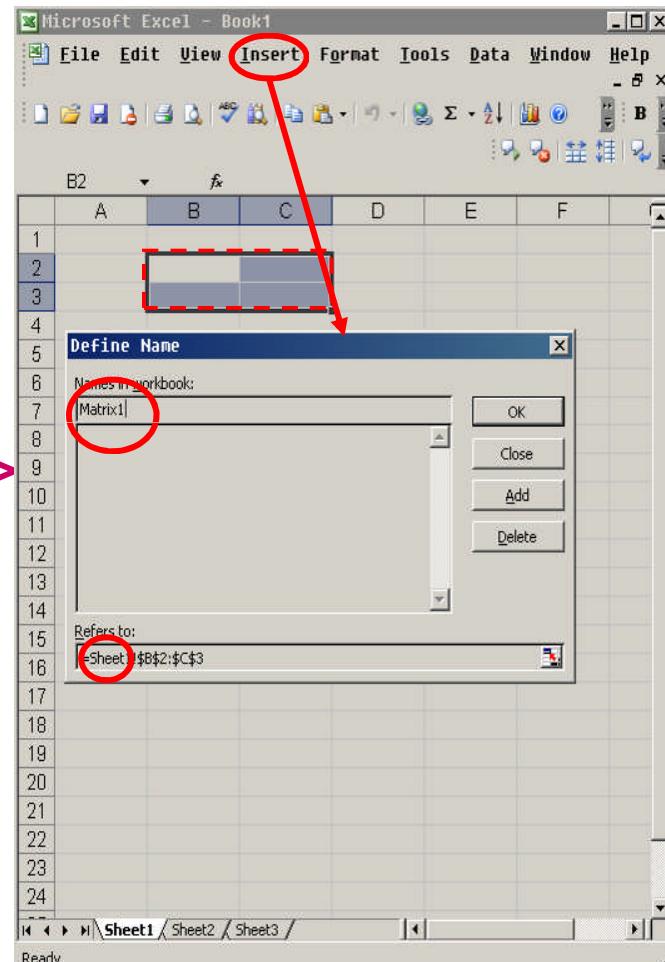


Naming group of Cells

- To assign a name to a range of cells, highlight it using the mouse and choose **Insert > Name > Define** and enter a name.
- Choose a useful name.
- Remember Excel does not distinguish between the names **MATRIX1**, **Matrix1** and **matrix1**.

Entering a Matrix

- Choose a location for the matrix (or vector) and enter the elements of the matrix.
- Highlight the cells of the matrix and choose **INSERT > NAME > DEFINE**.
- Enter a name for the matrix.
- You can now use the name of the matrix in formulae.



Matrix +- *

To add two **named** 3 x 2 matrices A and B:

Highlight a blank 3 x 2 results area in the spreadsheet. (If the results area is too small, you will get the wrong answer.)

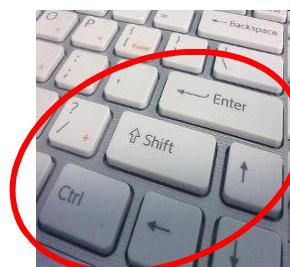
Type **=A+B** in the formula bar and press the **CONTROL-SHIFT-ENTER** keys simultaneously, to get the matrix result.



If you click on any cell in the result, the formula **{=A+B}** will be displayed, the **{ }** brackets indicate a matrix (array) command.

Matrix Transpose

- Suppose B is a 3×2 matrix.
- $B' = B$ transpose is a 2×3 matrix.
- Select a 2×3 results area, type **=TRANSPOSE(A)** in the formula bar and press **CTRL-SHIFT-ENTER**.



Matrix Multiplication

- Suppose A and B are named 3 x 2 and 2 x 3 matrices.
- Then A^*B is 3 x 3 and B^*A is 2 x 2. In general, $AB \neq BA$.
- Select a blank 3 x 3 area for the result A^*B .
- Type **=MMULT(A,B)** in the formula bar and press **CTRL-SHIFT-ENTER** to compute AB .



Exercise: Mmult

Make A[3x2] and B[2x3]

Check that $(AB)' = B'A'$, using
MMULT and TRANSPOSE

Using R Studio

```
A=matrix(sample.int(6,3*2,replace=1), 3, 2)
```

```
B=matrix(sample.int(6,3*2,replace=1), 2, 3)
```

```
t(A %*% B)
```

```
t(B) %*% t(A)
```

```
> A
 [,1] [,2]
 [1,] 1 4
 [2,] 4 1
 [3,] 3 3
> B
 [,1] [,2] [,3]
 [1,] 5 2 2
 [2,] 3 5 2
> t(A %*% B)
 [,1] [,2] [,3]
 [1,] 17 23 24
 [2,] 22 13 21
 [3,] 10 10 12
> t(B) %*% t(A)
 [,1] [,2] [,3]
 [1,] 17 23 24
 [2,] 22 13 21
 [3,] 10 10 12
```

Matrix Inverse and determinant

- Suppose B is a square 2×2 matrix.
- Select a 2×2 area for the inverse of B.
- Type **=MINVERSE(B)** in the formula bar and press **CRTL-SHIFT-ENTER**.
- Find $|B|$ =determinant of B using
=MDETERM(B)
- If determinant of B = $|B|=0$, it is not invertible.
- Let A and B be 3×3 , and $|A| \neq 0$, $|B| \neq 0$,
Compute and compare $(AB)^{-1} = B^{-1}A^{-1}$.

Linear Algebra in Excel



Microsoft
Excel 2003

Excel Grid



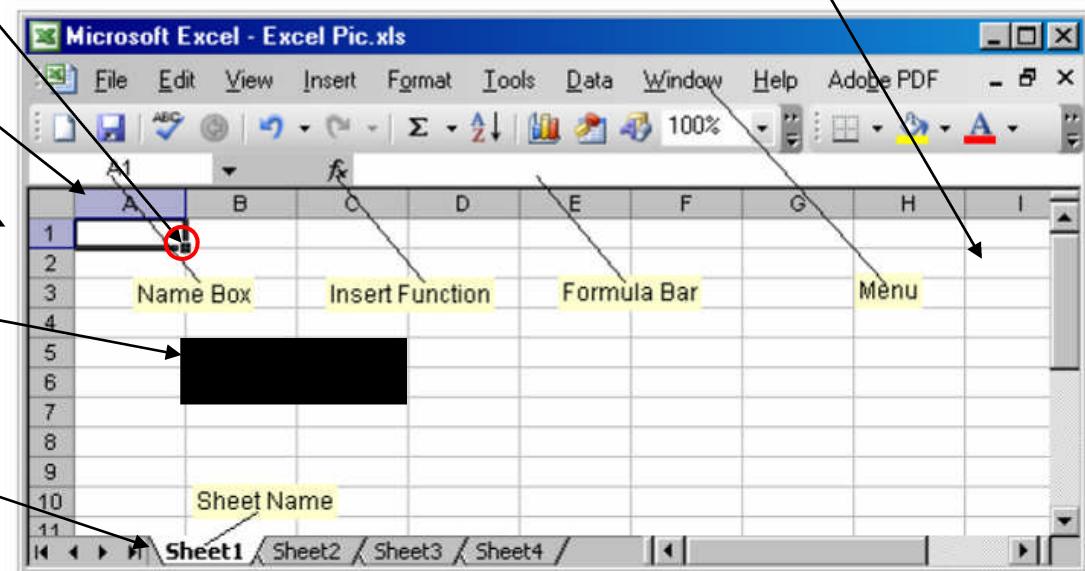
- Excel file have extension .xls or .xlsx (2007).
- Work book is a 2D matrix/grid of **CELLs** containing data
- Cells are then named by Column+Row, e.g. A1, B5, I2
- Draggable corner to copy cells

Columns are: A,B,C,D

Rows are: 1,2,3,4,...

Matrix: B5:B6;C5:C6

Sheets: 1,2,...



Solving Ax=b

$$\begin{array}{rcl} x + 2y + z & = & 0 \\ y - z & = & 2 \\ x & + & 2z = 1 \\ \hline \end{array}$$

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & -1 & 2 \\ 1 & 0 & 2 & 1 \end{pmatrix}$$

Write system of equations in matrix form.

$$\begin{array}{rcl} x + 2y + z & = & 0 \\ y - z & = & 2 \\ -2y + z & = & 1 \\ \hline \end{array}$$

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & -1 & 2 \\ 0 & -2 & 1 & 1 \end{pmatrix}$$

Subtract first row from last row.

$$\begin{array}{rcl} x + 2y + z & = & 0 \\ y - z & = & 2 \\ -z & = & 5 \\ \hline \end{array}$$

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & -1 & 2 \\ 0 & 0 & -1 & 5 \end{pmatrix}$$

Add 2 copies of second row to last row.

Now solve by back-substitution: $z = -5$, $y = 2-z = -3$, $x = -2y-z = 11$

Using Excel

1. Enter the numbers (A, b),
2. Name the matrices (A) and arrays (x,b)
3. Call math and matrix functions on your data to compute x.
 $x = A^{-1}b$
4. Compute A^{-1}

	A	B	C	D	E	F	G	H
1								
2		1	2	1	x	11	0	
3	A		0	1	-1	-3	2	
4		1	0	2	-5	1		
5	DetA	-1						
6		-2	4	3				
7	AINV	1	-1	-1				
8		1	-2	-1				
9								

Microsoft Excel - Book1

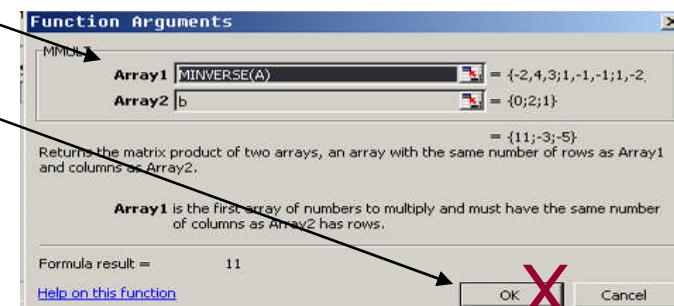
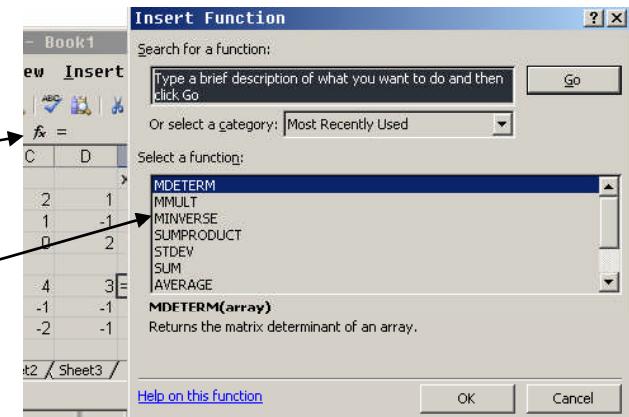
File Edit View Insert Format Tools

=MMULT(MINVERSE(A),b)

Sheet1 Sheet2 Sheet3

Linear Algebra in Excel

- Type in your matrix into Excel cells
- Select matrix and give it a name, e.g. 'A', 'b', 'x'.
- Select the cells you want the result in.
 - click on 'fx' to enter a formula for the result,
 - find your function in the list
 - type in the symbolic arguments,
 - press **Control-Shift-Enter**, not 'OK'.
if the result is a matrix.



Solving Ax=b in Excel

- Select x (E2:E4), click on fx
- Compute $x = \text{MINVERSE}(A) * b$
- press **Control-Shift-Enter** instead of ok.



Microsoft Excel - Book1

File Edit View Insert Format

MDETERM ▾ X ✓ A =MMULT(MINVERSE(A),b)

	A	B	C	D	E	F
1				x	b	
2		1	2	1	A(b)	0
3	A	0	1	-1	-3	2
4		1	0	2	-5	1
5	DetA	-1				
6		-2	4	3		
7	AINV	1	-1	-1		
8		1	-2	-1		
9						

Function Arguments

MMULT

Array1 MINVERSE(A)

Array2 b

Returns the matrix product of two arrays, an array with the same number of rows as Array1 and columns as Array2.

Array1 is the first array of numbers to multiply and must have the same number of columns as Array2 has rows.

Formula result = 11

Help on this function

OK X Cancel

The screenshot shows an Excel spreadsheet with data in cells A1:F9. Cell E2 contains the formula =MMULT(MINVERSE(A),b). A callout points from the text "Control-Shift-Enter instead of ok." to the "OK" button in the Function Arguments dialog box. Another callout points from the text "Control-Shift-Enter instead of ok." to the Control key on a keyboard image. The Function Arguments dialog box shows "Array1" set to "MINVERSE(A)" and "Array2" set to "b". The formula result is displayed as 11.

Command Line shells

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

CMD Command line

Windows Disk consist of drives:

C:\ D:\ E:\

And each disk has directories, e.g.

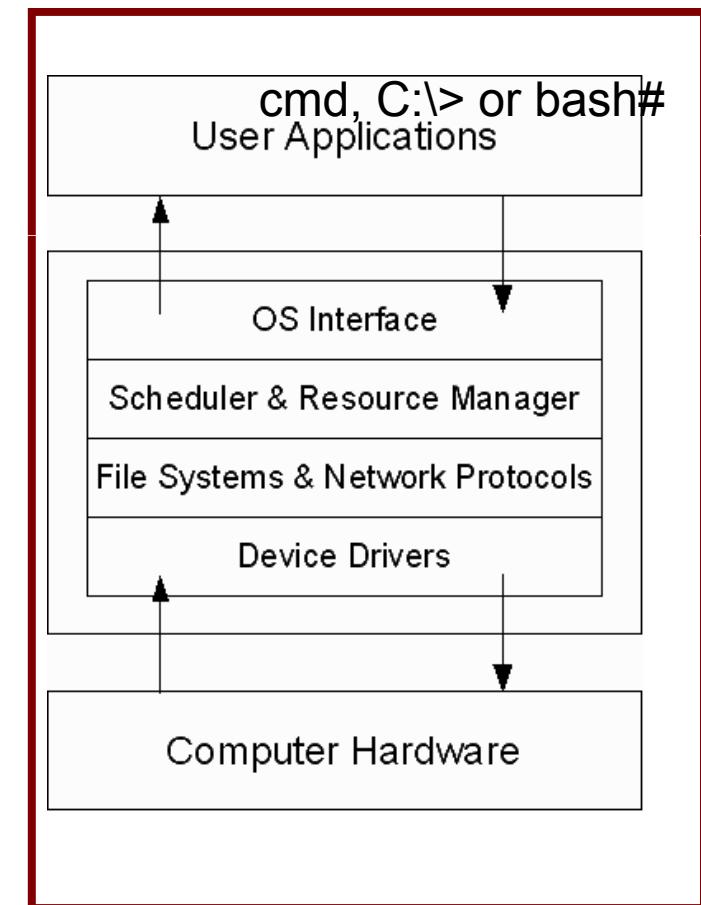
C:\windows and

C:\Documents and Settings\...\Desktop

And each directory contains files.

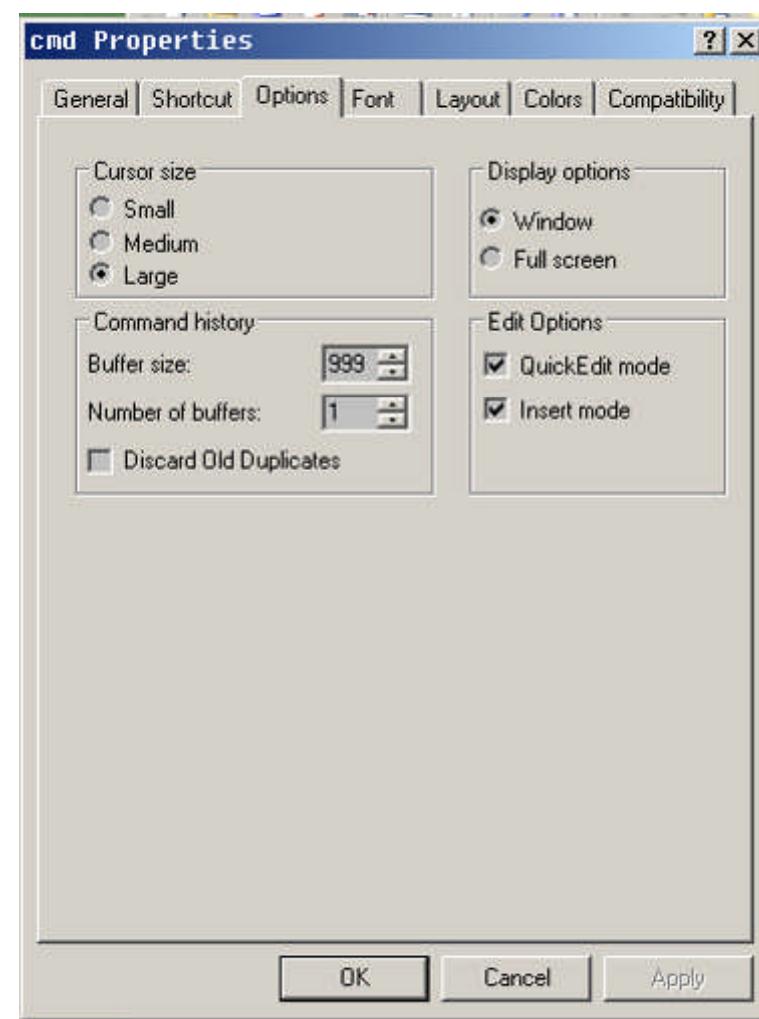
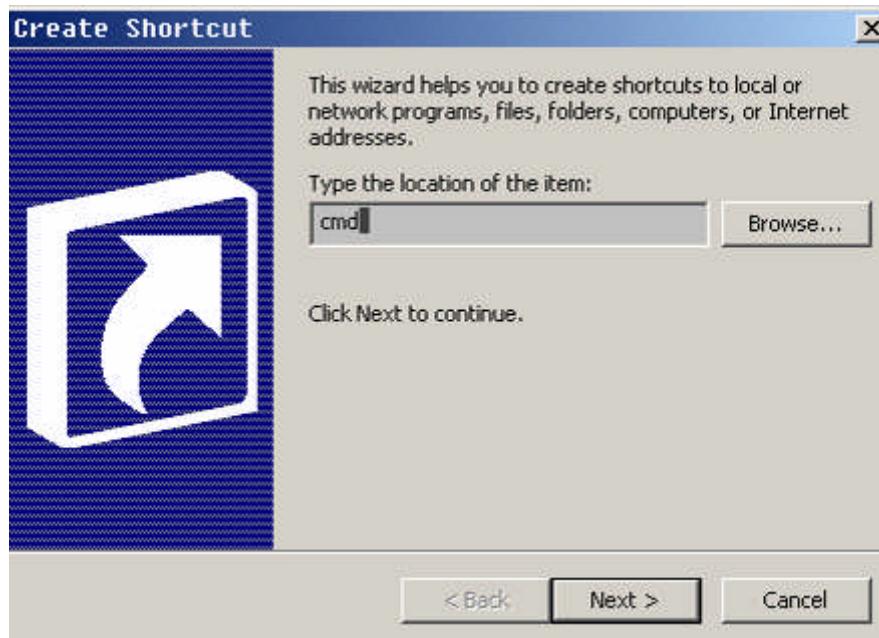
Files have a name and extension,
e.g.

C:\WINDOWS\system32\system.ini

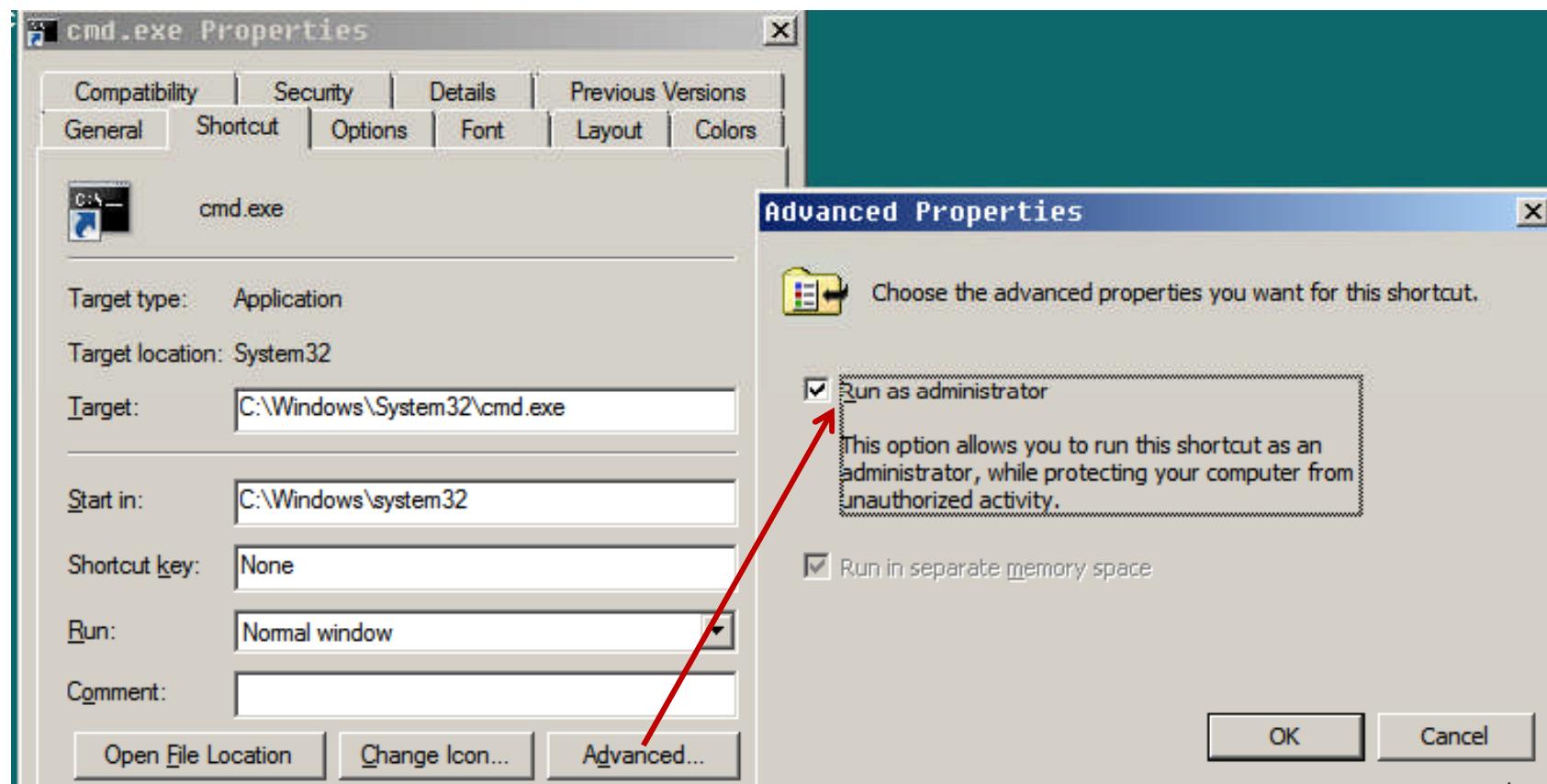


Create shortcut for cmd

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



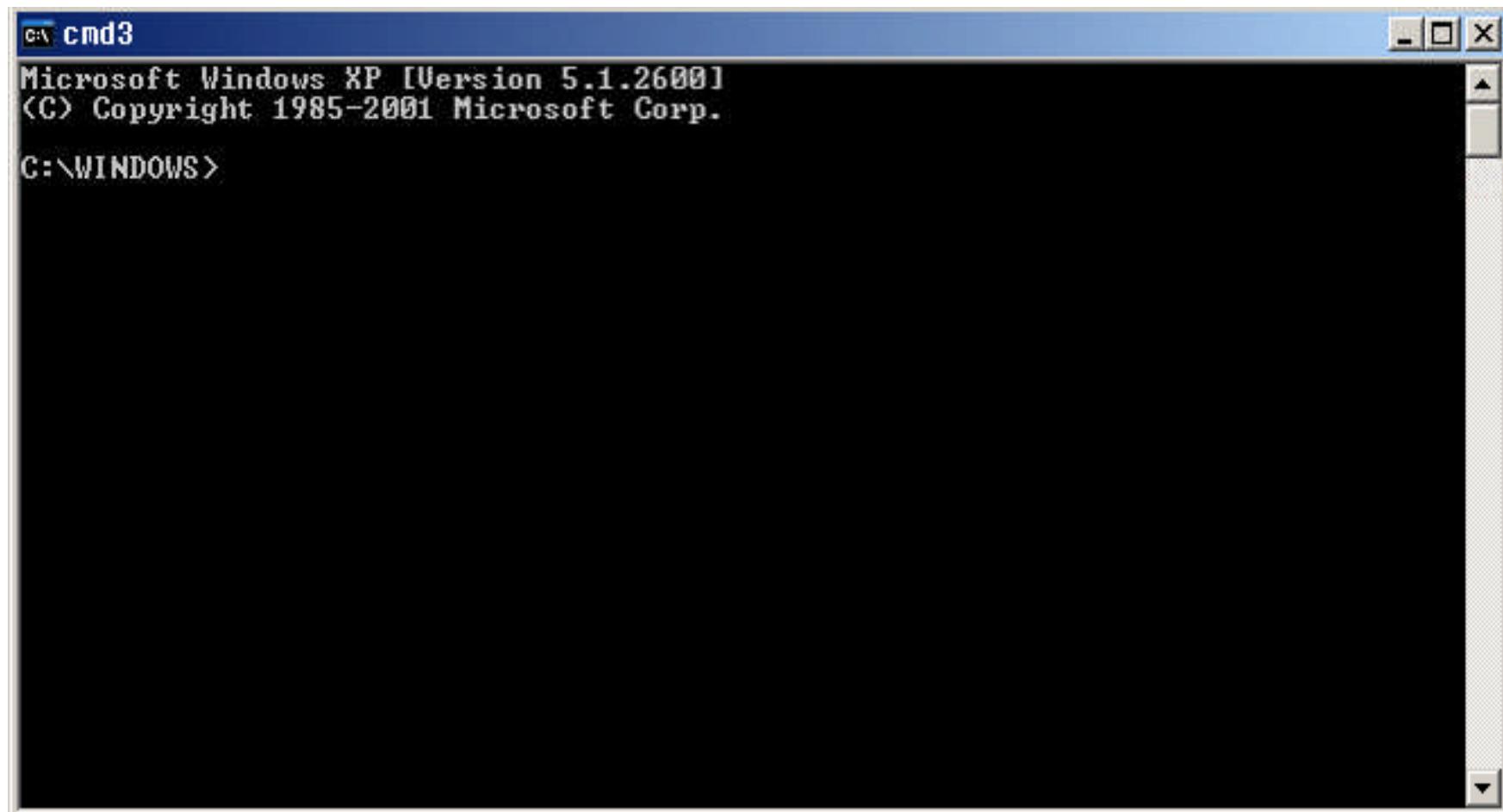
Always run cmd as admin



Cmd (console) window

Only text, no graphics

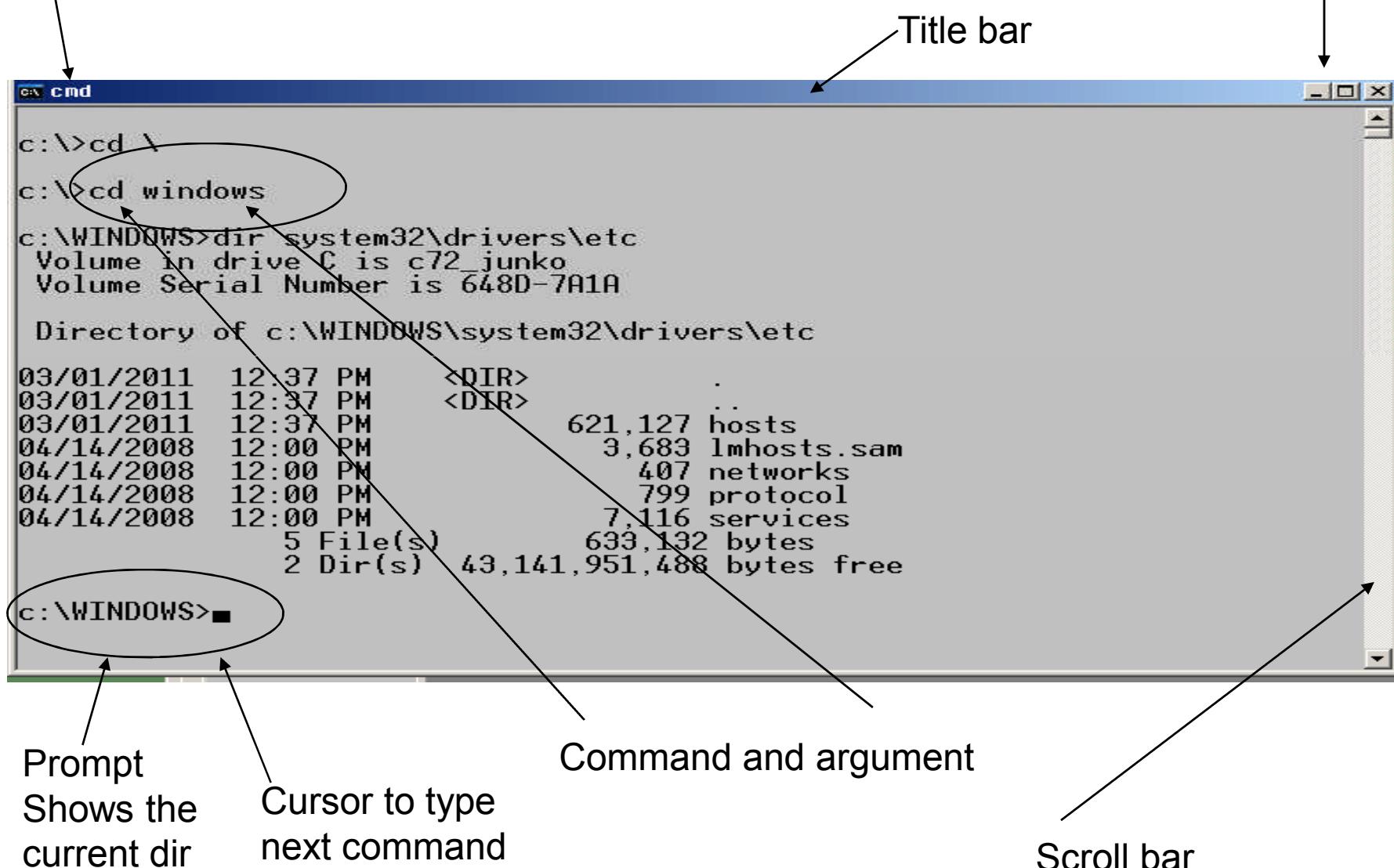
- Start -> Run -> cmd



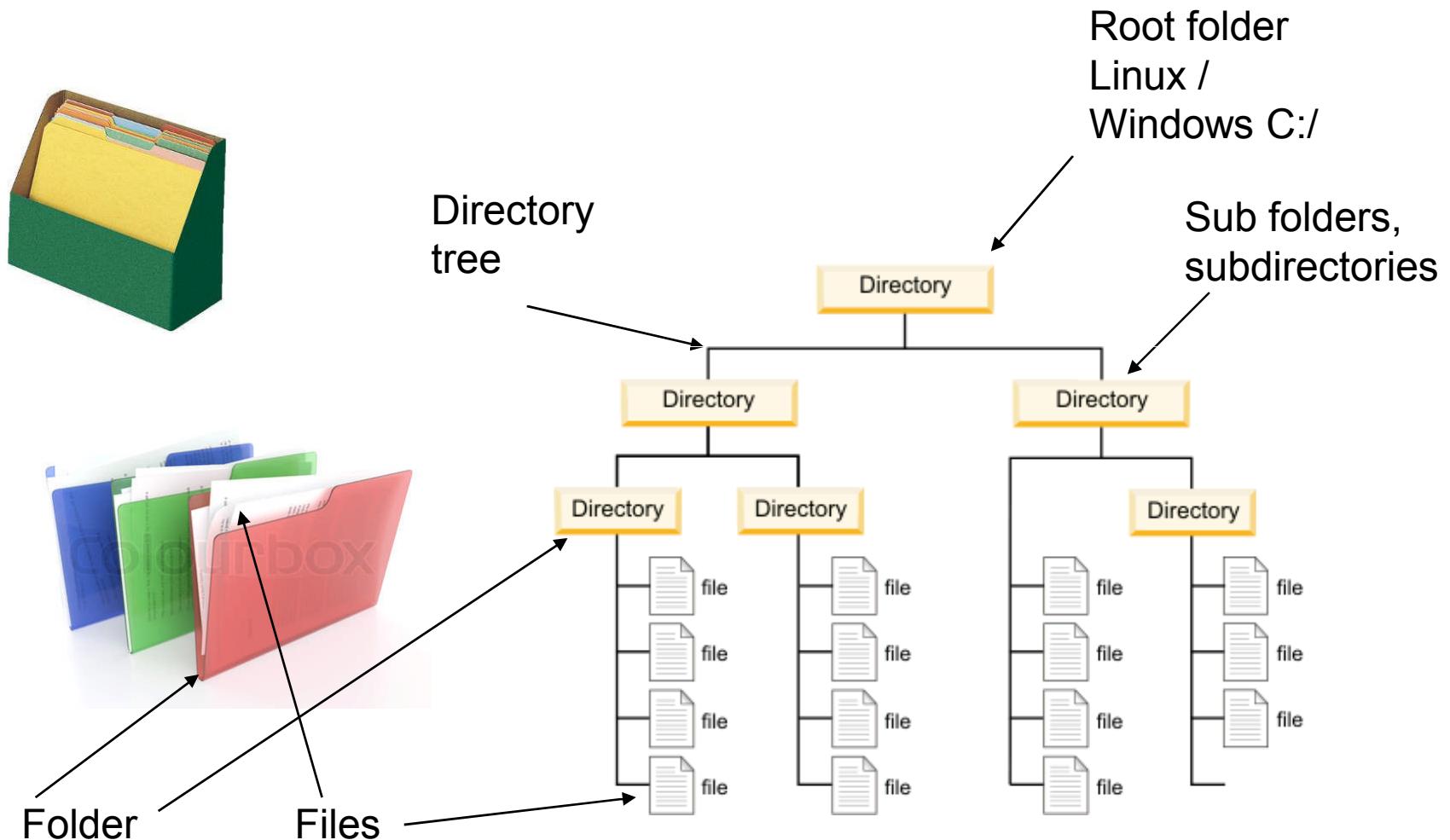
Cmd icon
and name

cd and dir commands

Buttons to
- Minimize
[] Maximize
X Close window



Folders and files



Folder cabinets (File System)

C: ← Primary Disk volume

C:\ ← Root directory

C:\windows\ ← Windows Folder

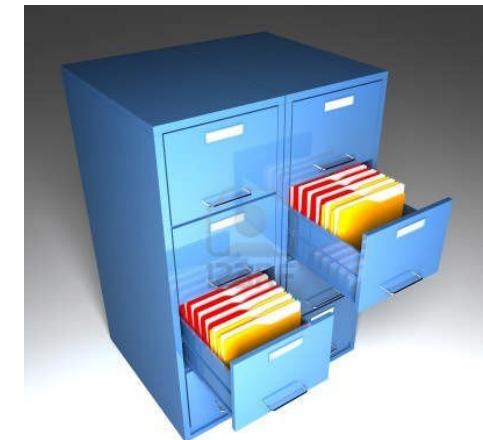
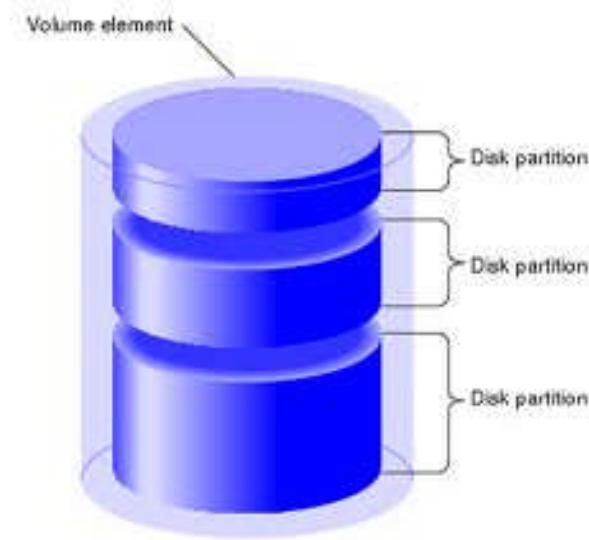
C:\Document and Settings\

D: Secondary Disk

E: DVD drive

F: USB disk

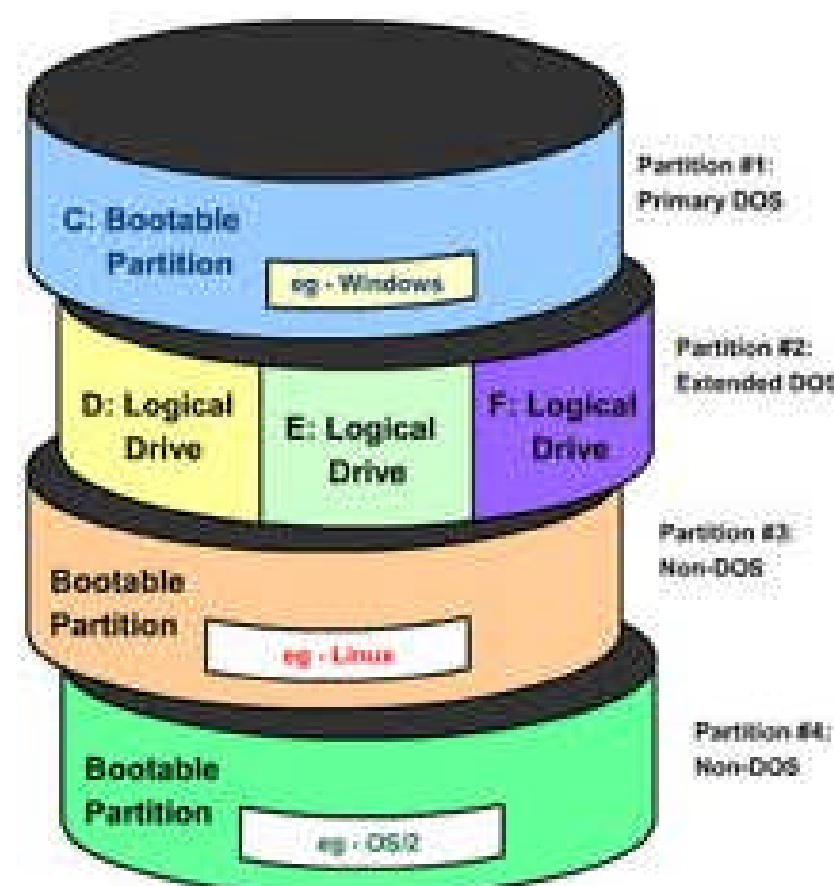
Filesystems: Fat32, NTFS, ext2, etc



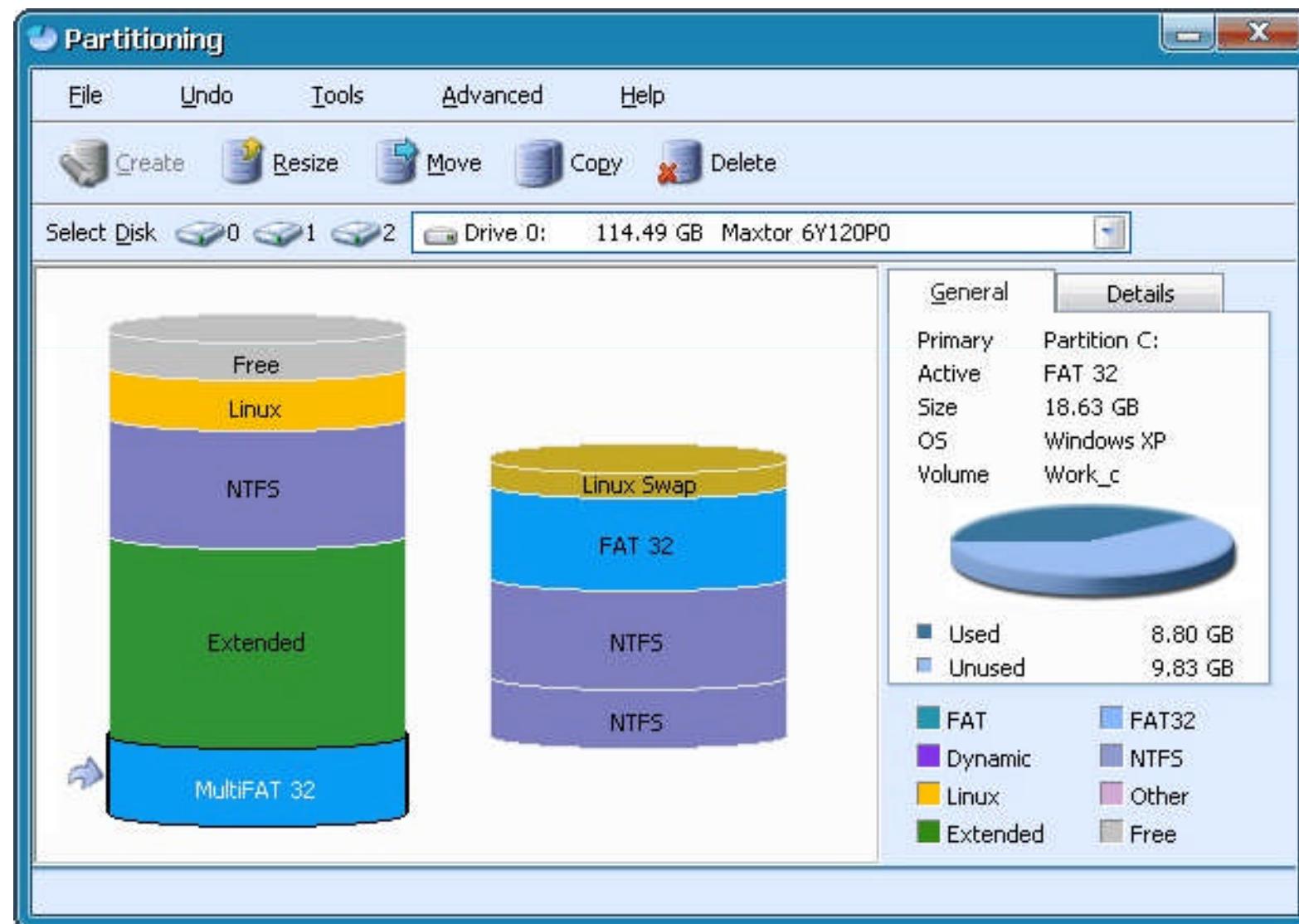
Disk volumes

`/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'

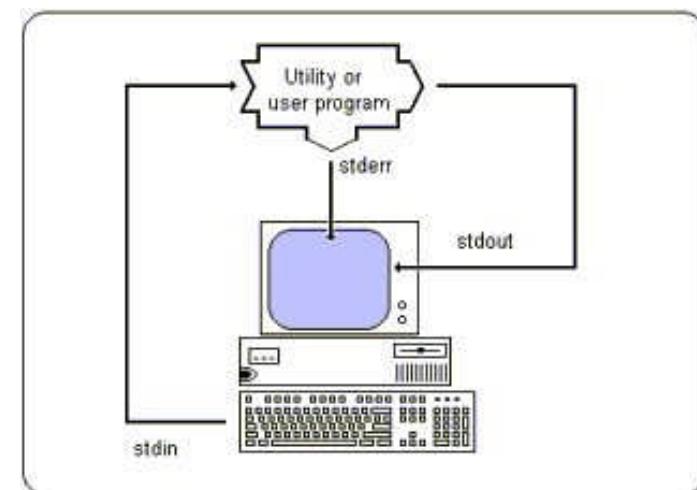
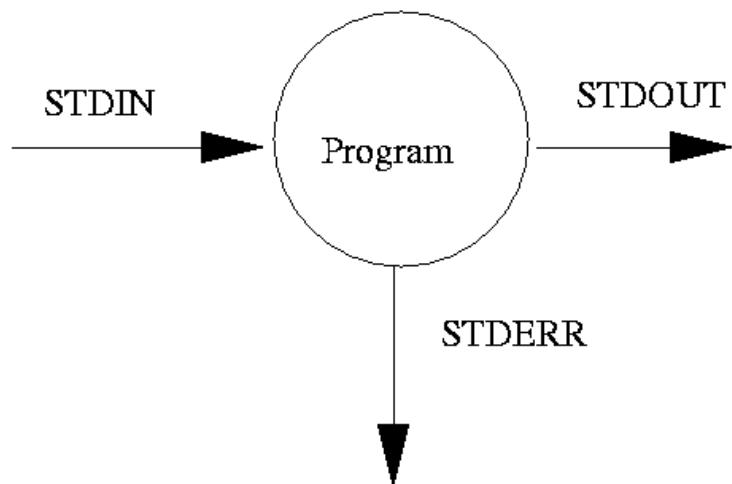
```
C:\> doskey ls=dir $*
```

```
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 tips and tricks

command completion

1. Use *Clink* for Bash style completion; and
ConEmu for multi tab consoles.
(google search for *clink* and *conemu cmd*)
2. Use Hat (^) for quoting and line-continuation, e.g.
c:\> grep ^# *.sh ^
| wc

MkLink - symlinks

```
> 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

Cmd Tips - alias

Some of cmd's (date, echo, find, mkdir) don't work as well as the unix, but you can override them with doskey aliases:

```
> @doskey date=c:\cygwin64\bin\date.exe $*
> @doskey echo=c:\cygwin64\bin\echo.exe $*
> @doskey mkdir=c:\cygwin64\bin\mkdir.exe $*
> @doskey find=c:\cygwin64\bin\find.exe $*
```

```
> date
```

```
Tue Sep 19 11:55:45 IST 2017
```

Cmd tips - `external cmd`

Get Output of external command from a cmd script

```
c:\> cat some.cmd
```

```
    :: Set MYDATE=2016-09-09 from output of gnu date
```

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

```
echo "Today is %MYDATE%"
```

```
c:\> some.cmd
```

```
"Today is 2017-09-19"
```

- * Use cygwin / bash / perl for more complicated scripting on windows. They are easier to test/maintain/portable.

Useful Tools on Windows

1. Cygwin, bash, perl, gcc
2. Notepad++, gvim
3. Google Chrome
5. Java OpenJDK (not Oracle-Java) and Idea Ide.
6. Xampp (web server, apache, mysql, php)
7. Python 2.7
8. IrfanView, VLC
9. audacity, 1by1
10. Gcc and CodeBlocks
11. MS VC++ Ide

Protecting windows

1. Msconfig – disable 3rd party programs from updating/auto starting.
2. ProcessExplorer, to find rogue programs.
3. TcpView, to find rogue tcp/ip connections.
4. Blacklisted hosts file, see winhelp2002.mvps.org
5. Disable IE and Chrome plugins



Bash shell

by moshahmed@gmail.com NITK 2013

Bash shell

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

Windows users can download
c:/cygwin/bin/bash with cygwin.

History:

sh (ATT unix) → ksh → **bash** (current)

Obselete: zsh, csh, tcsh (bad design)

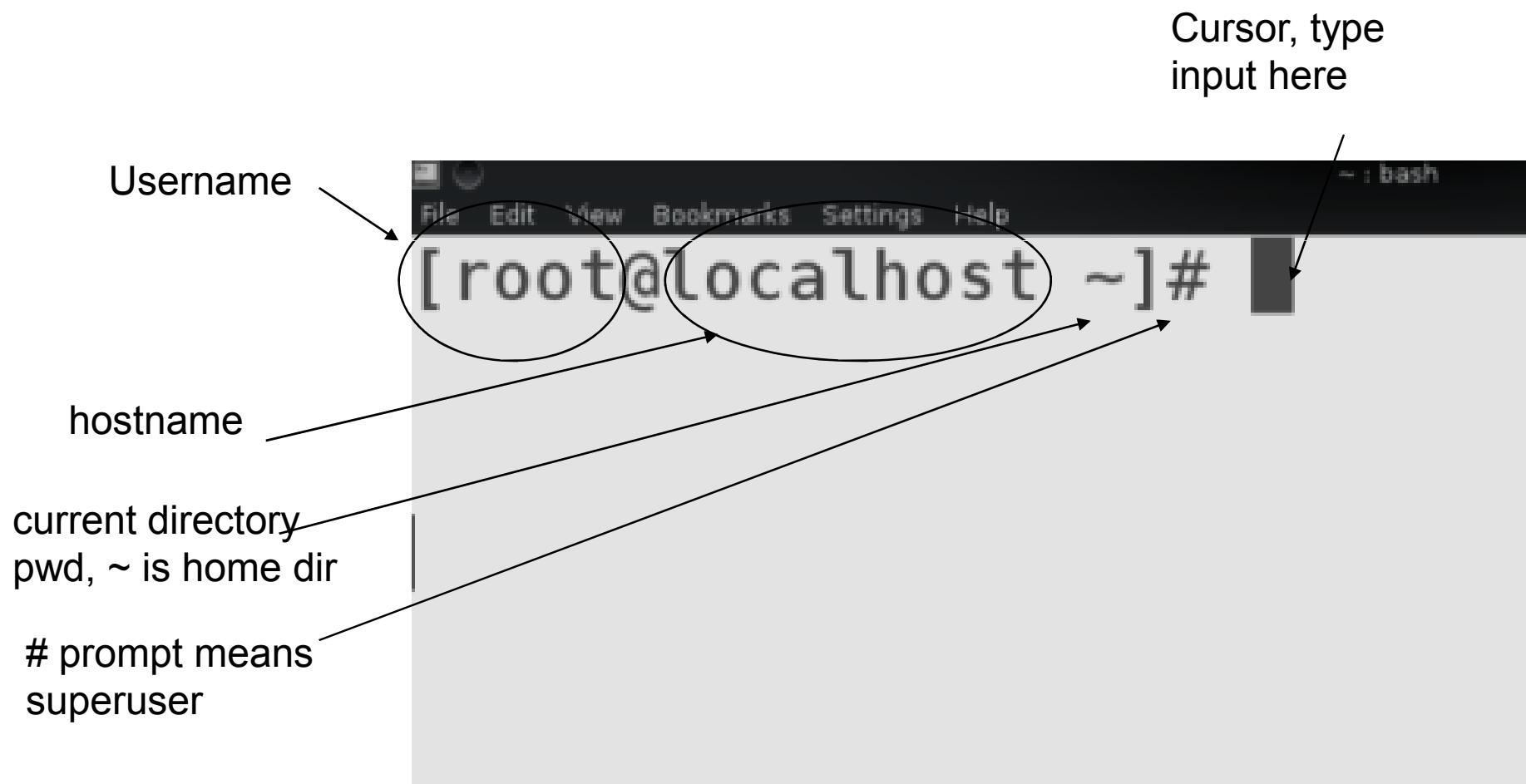


Windows Cygwin Setup

- 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
- \$

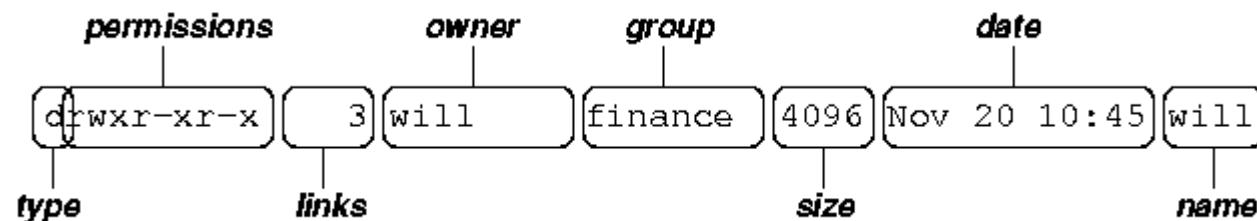
Start bash in a terminal

- start > terminal



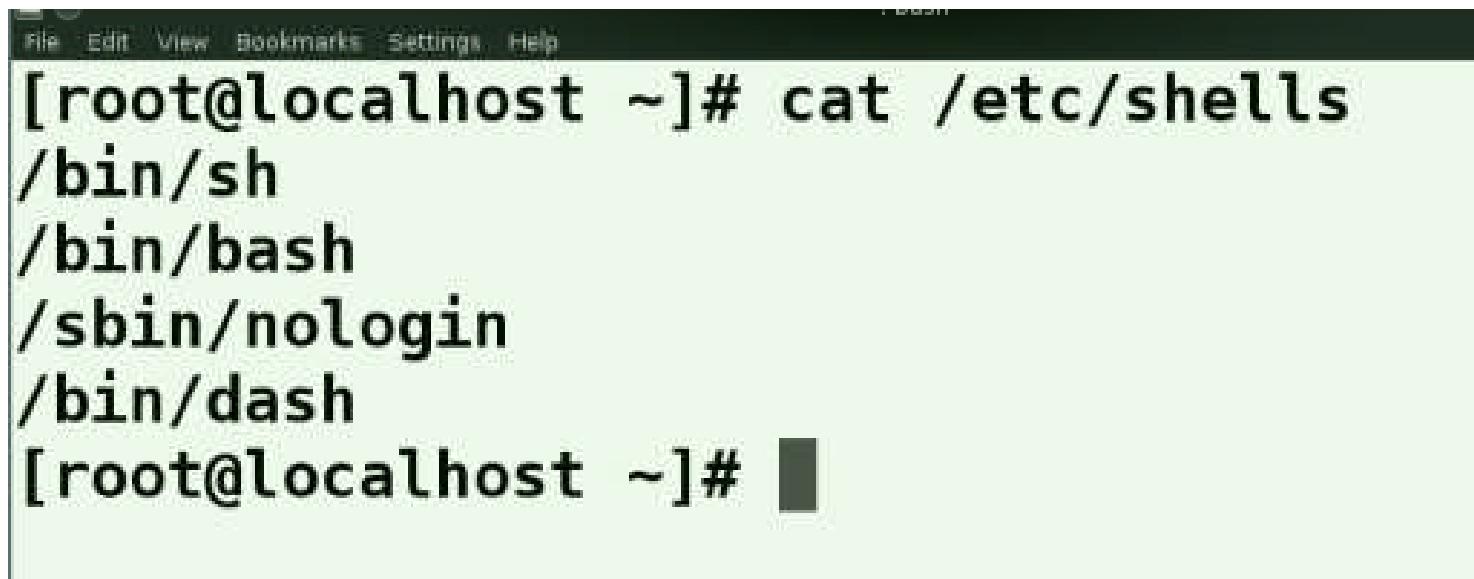
ls – list directory and files

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.



The image shows a terminal window with a dark header bar containing menu items: File, Edit, View, Bookmarks, Settings, Help. The main area of the terminal displays the command [root@localhost ~]# cat /etc/shells followed by a list of shell paths: /bin/sh, /bin/bash, /sbin/nologin, and /bin/dash. The prompt [root@localhost ~]# is visible at the bottom, along with a black vertical bar representing the cursor.

```
[root@localhost ~]# cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/dash
[root@localhost ~]#
```

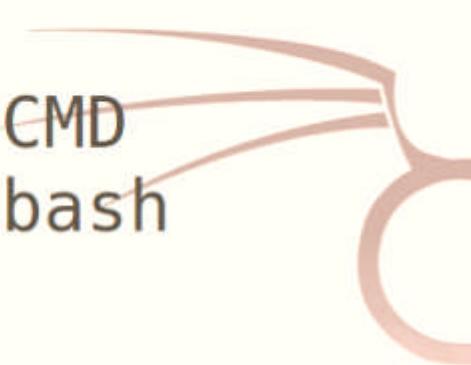
Environment

```
# echo $HOME
```

```
# ps -p $$ .. process info
```

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

TTY .. is the controlling terminal of this bash



```
root@bt:/# echo $SHELL
/bin/bash
root@bt:/# ps -p $$
```

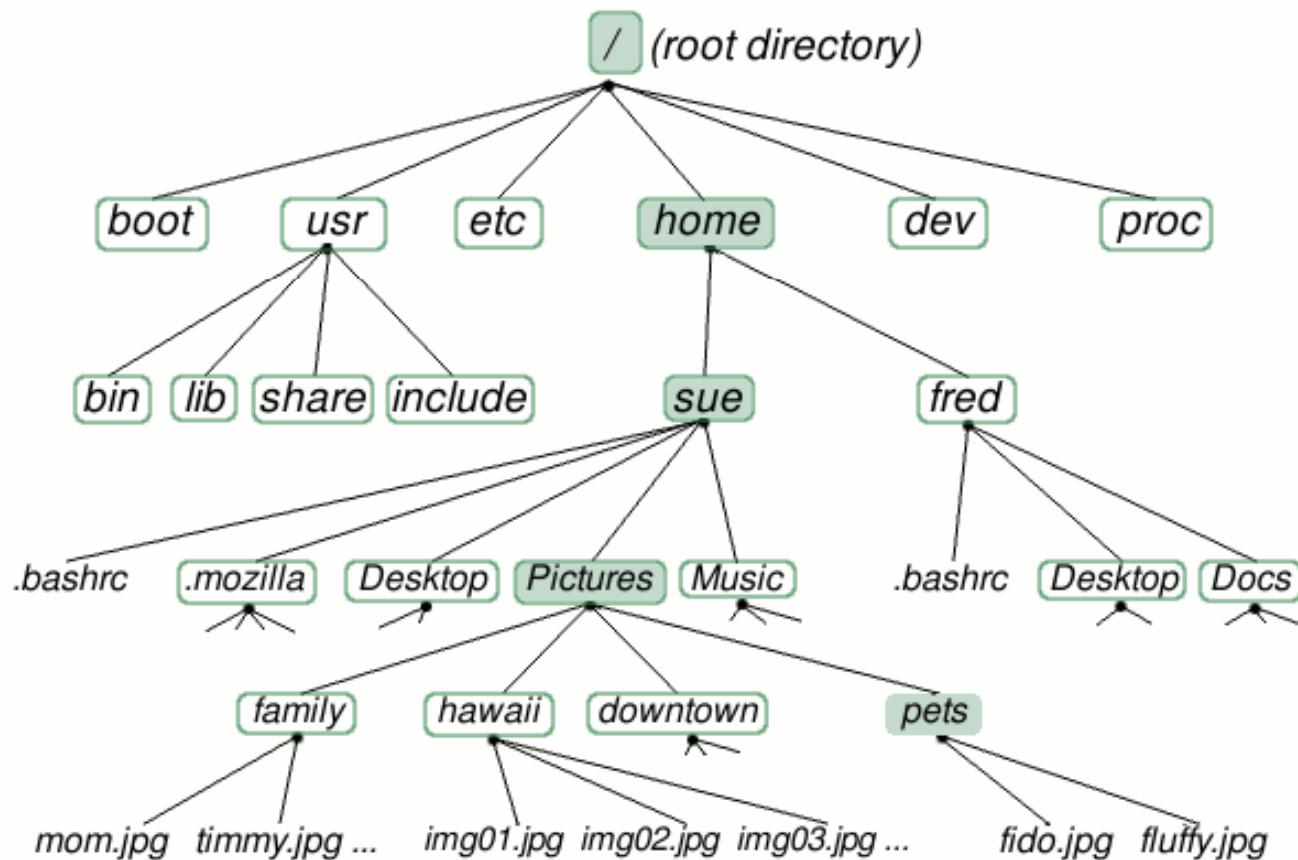
PID	TTY	TIME	CMD
4481	pts/0	00:00:00	bash

```
root@bt:/#
```

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.

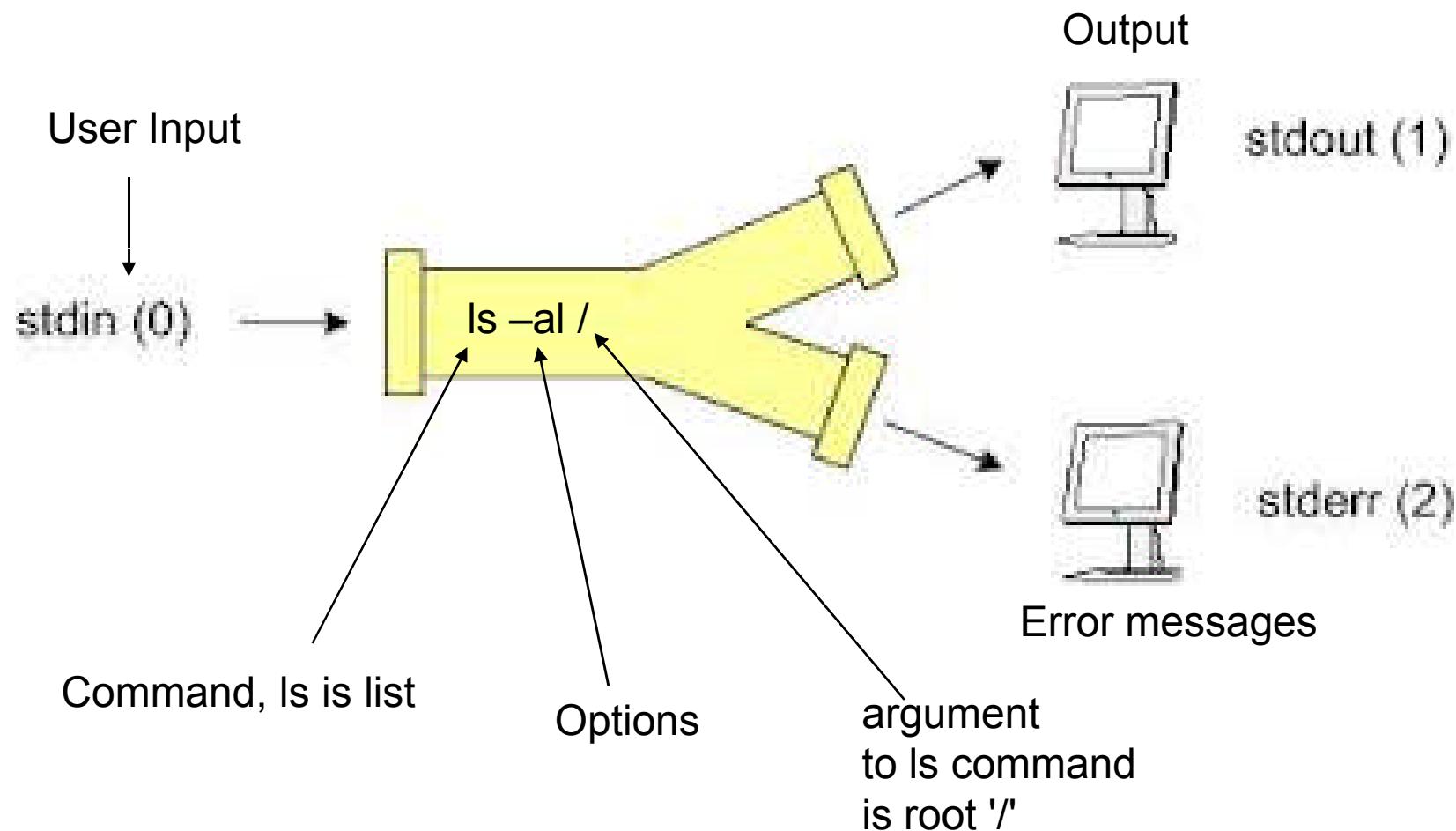
More Unix commands

- **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 (stderr 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)

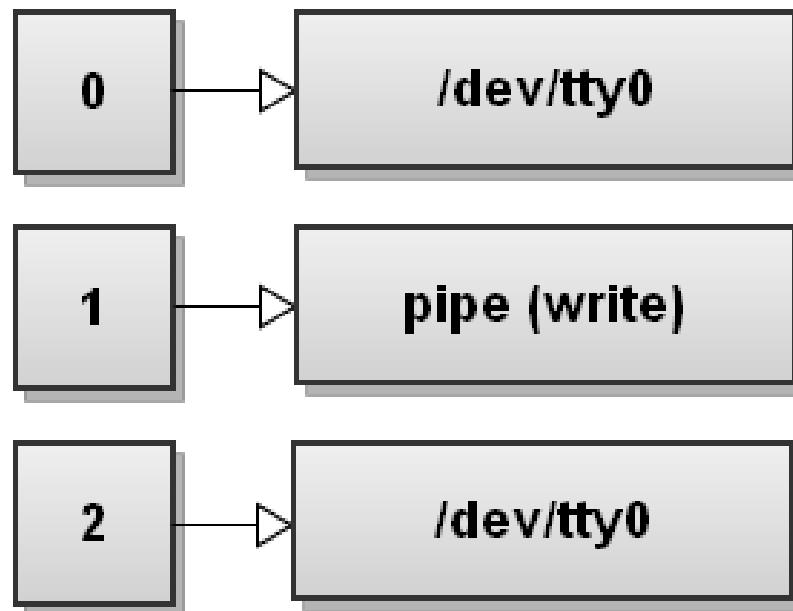
Save output and error messages of gcc, send stdout to file x, and also redirect stderr/2 to stdout/1.

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

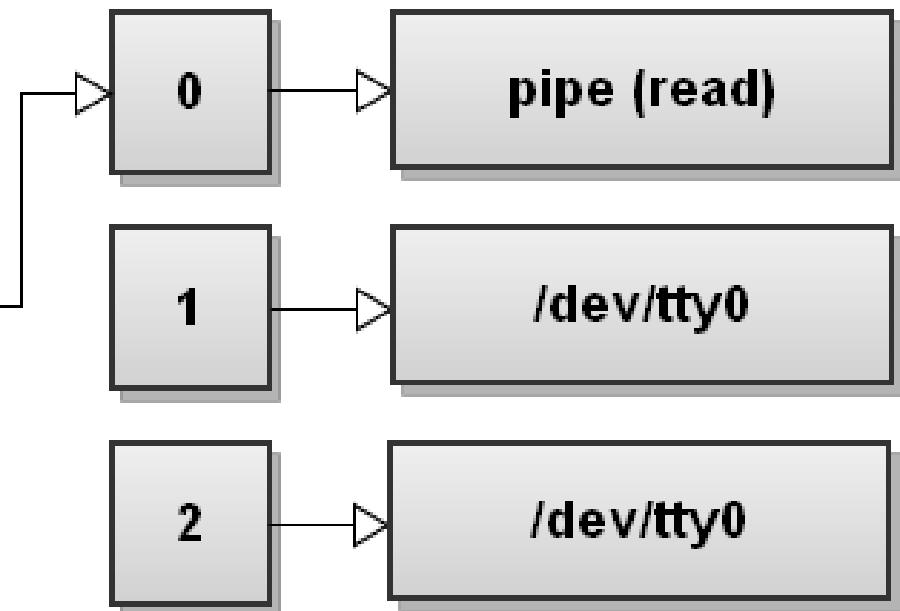
pipe, and io redirection

\$ command1 | command2

command1's file descriptors



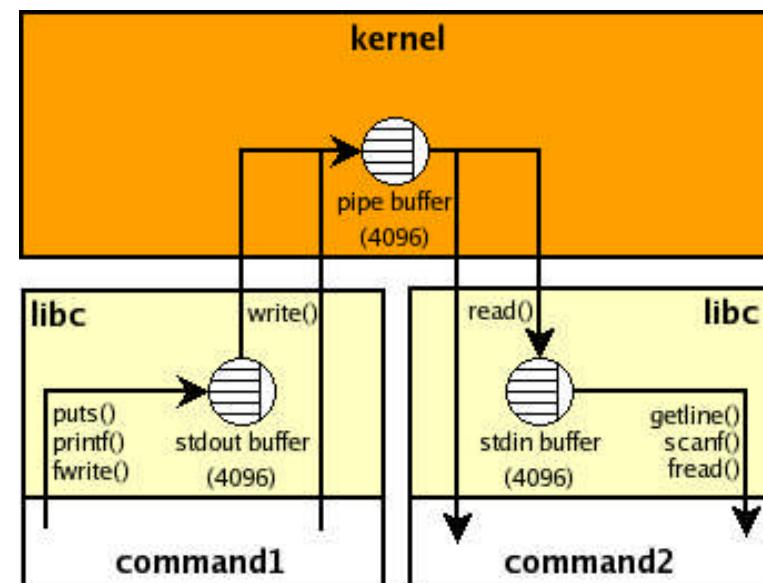
command2's file descriptors



Piping ‘|’

Pipe output of first cmd
to next cmd, example

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

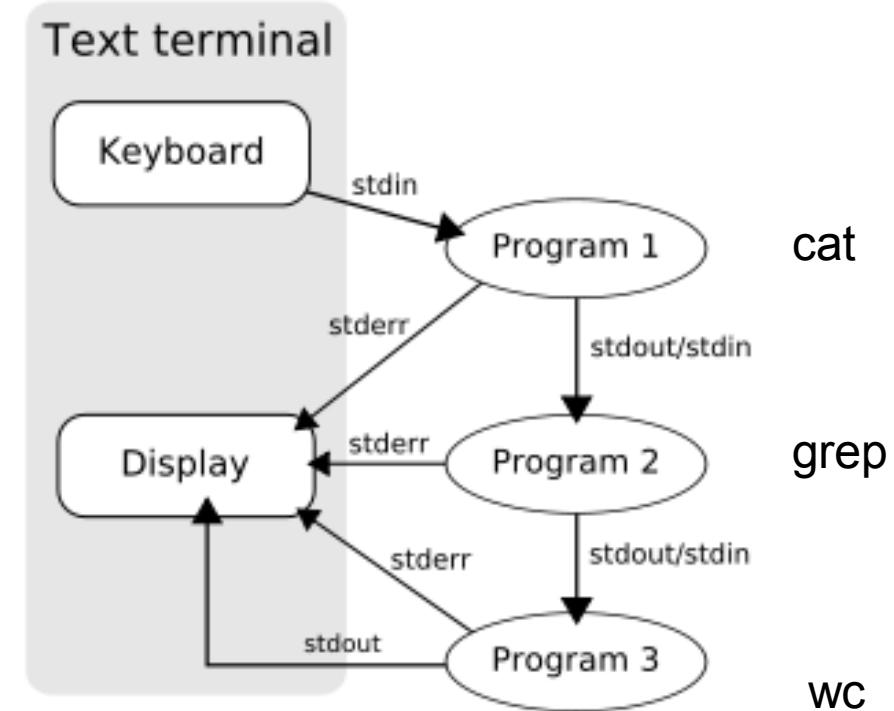


Pipe example with 3 commands

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”  
;elseif [[ 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
File separator	\	/
Root	C:\	/
Line ending <small>(Fix dos2unix, unix2dos)</small>	\r\n	\n
Shell	cmd.exe	bash
File case	dir == DIR	ls != LS
Syntax	Inconsistent	Good
variables	%USER%	\$USER

Windows commands with same names as Unix commands:

- echo, find, date, mkdir, link, time (for if)
- On windows, use doskey for aliasing, e.g.
- @doskey find=c:\cygwin64\bin\find.exe \$*

Other tools:

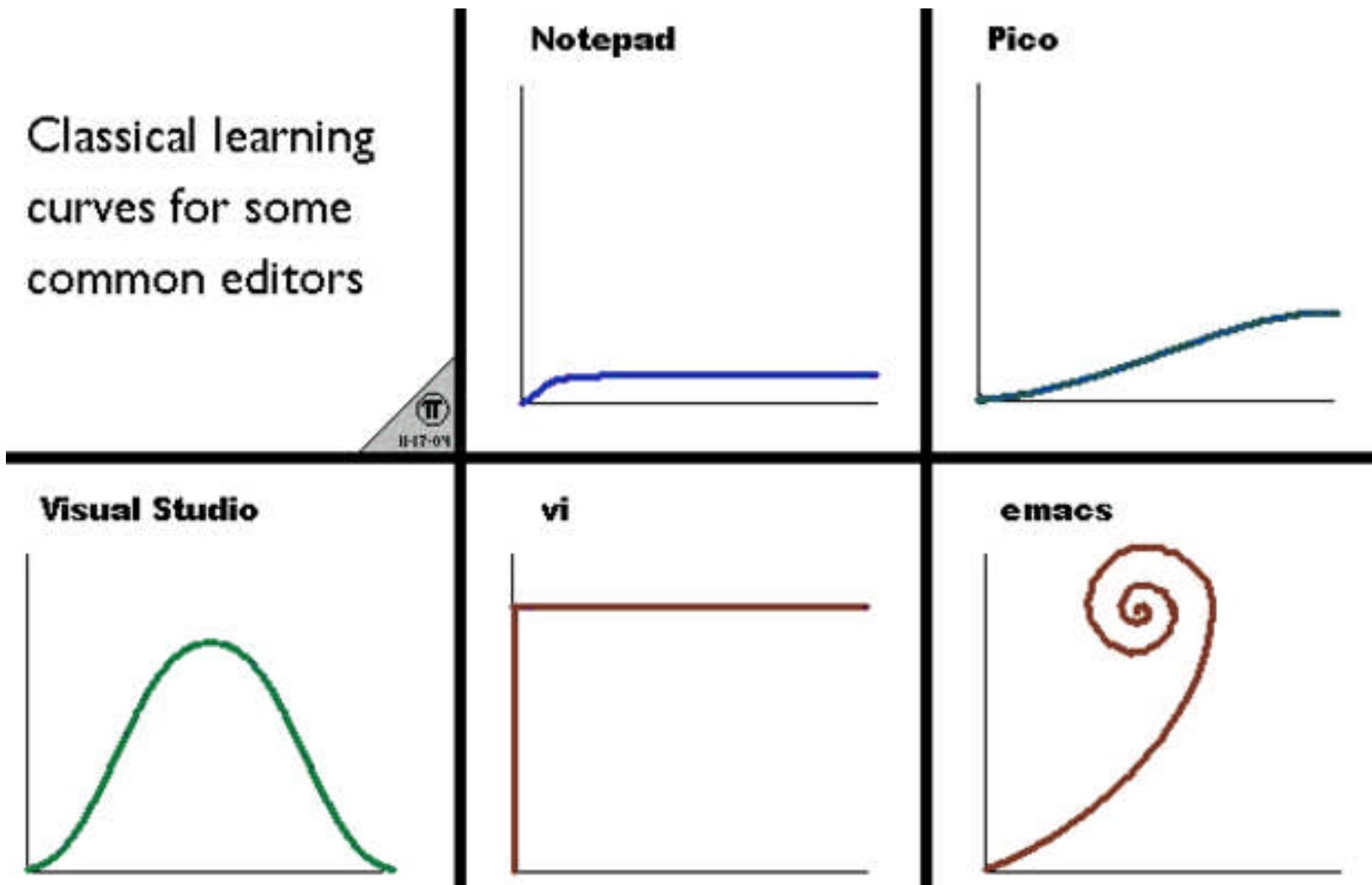
- ssh (secure shell)
- putty (ssh from windows to unix)
- tmux/ gnu screen (virtual terminal)
- find, grep,

Scripting languages:

- perl, python3, nodejs

Editing

Emacs, Vim, Notepad++



Vim

- ed > ex > vi > vim > gvim
- Same command line available on all Unix & Windows machines since 1970s.
- Single letter commands
- Regexp > extended RE > PCRE.

CVS



- SCM: Source Code Management

SCM



PERFORCE
Version everything.



Old: cvs, perforce, svn, hg, fossil

New: git

Concepts in tracking files

- Repository – Backup copy of all files and versions.
- Head – current version file.
- Branch – alternate version of file.
- Operations
 - checkout, checkin, commit, review
 - diff, versions, conflicts, resolve, merge.
 - add, stash, revert, remove
 - push, pull

CVS

- CVS is *Concurrent Version System*
- Obsolete, but useful for single user
- To maintain backups of text files
- Works on windows and unix
- Free software.



CVS

Windows: Install cygwin-cvs or
wincvs or tortoise-cvs

Local

- `export CVSROOT=/path/to/cvsroot`

Remote

- `CVSROOT=:ext:user@server:/repo/`
 - `export CVS_RSH=ssh`
- `> cvs --help`

New local Repo for ~/src

```
> cvs -d c:/cvs/repo init  
> export CVSROOT=c:/cvs/repo  
> cd ~/src  
> cvs import src your-name begin  
> rm -rf ~/src # delete old copy  
> cvs co src # checkout new src from cvs
```

Using cvs

```
> vi file.c  
> cvs diff file.c  
> cvs status  
> cvs commit -m "My fixes" file.c
```

```
> cvs up      # Get other people's changes  
> vi file.c  # Fix conflicts, merge changes  
> cvs commit -m "merged"
```

Using cvs

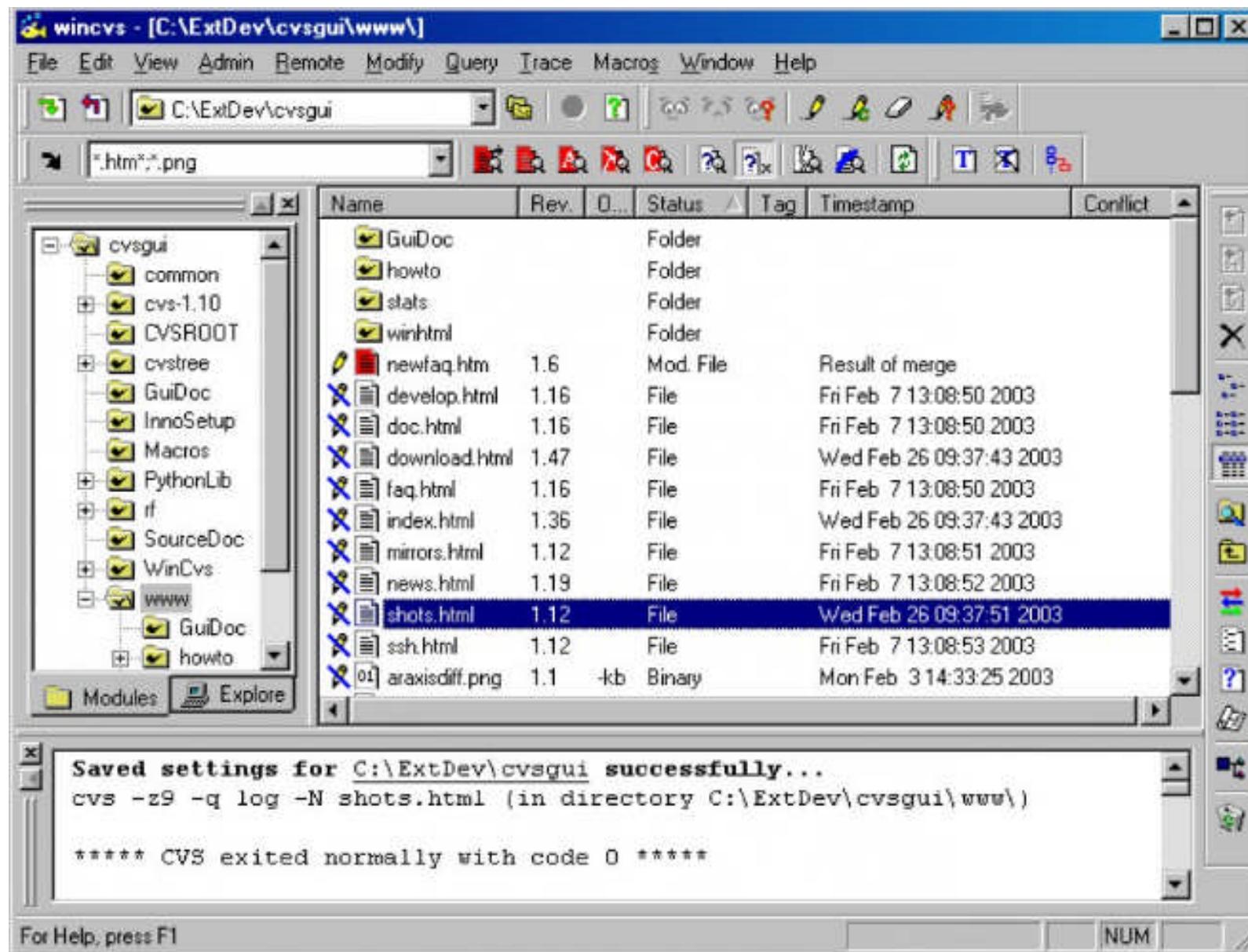
```
> cvs add new.h  
> cvs rm old.h  
> cvs commit -m "new files"  
  
> cvs log file.c # See the file history  
> cvs ann file.c # See who changed lines  
> cvs diff -r 1.1 file.c # See what changed
```

cvs tags

```
> cvs tag -R "Version1" src  
> cvs tag -b Branch2 src
```

WinCVS

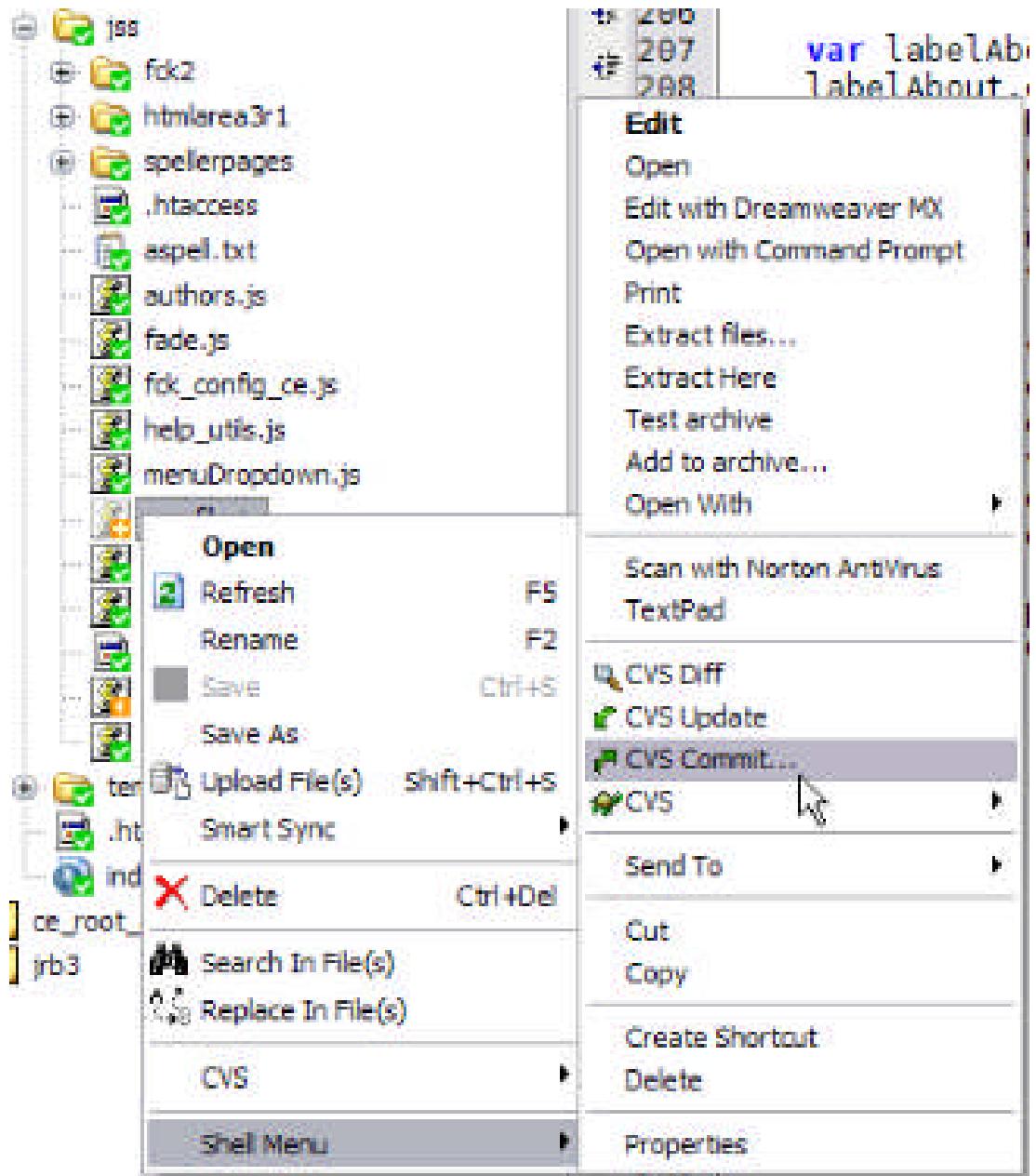
- Graphical GUI for CVS, useful for branches.



Tortoise CVS



- Use directly from Windows Explorer menus
- Also tortoise-git, tortoise-svn, ...



References

1. CVS <http://www.fisica.uniud.it/~glast/sw/cvs/cvsintro.html>
2. GIT <https://git-scm.com/>

GIT



- SCM: Source Code Management

SCM



PERFORCE
Version everything.



Old: cvs, perforce, svn, hg, fossil

New: git

Concepts in tracking files

- Repository – Backup copy of all files and versions.
- Head – current version file.
- Branch – alternate version of file.
- Operations
 - checkout, checkin, commit, review
 - diff, versions, conflicts, resolve, merge.
 - add, stash, revert, remove
 - push, pull

CVS vs Git

- CVS: old 1990s, GPL,
 - tracking changes in individual files
 - good for personal repo, single office.
 - branching hard
- Git : new 2005, GPL,
 - tracking Changes
 - good for distributed workers, sharing code
 - distributed
 - pushing CL from repo to repo
 - branching is fast. used by linux, github, opensource

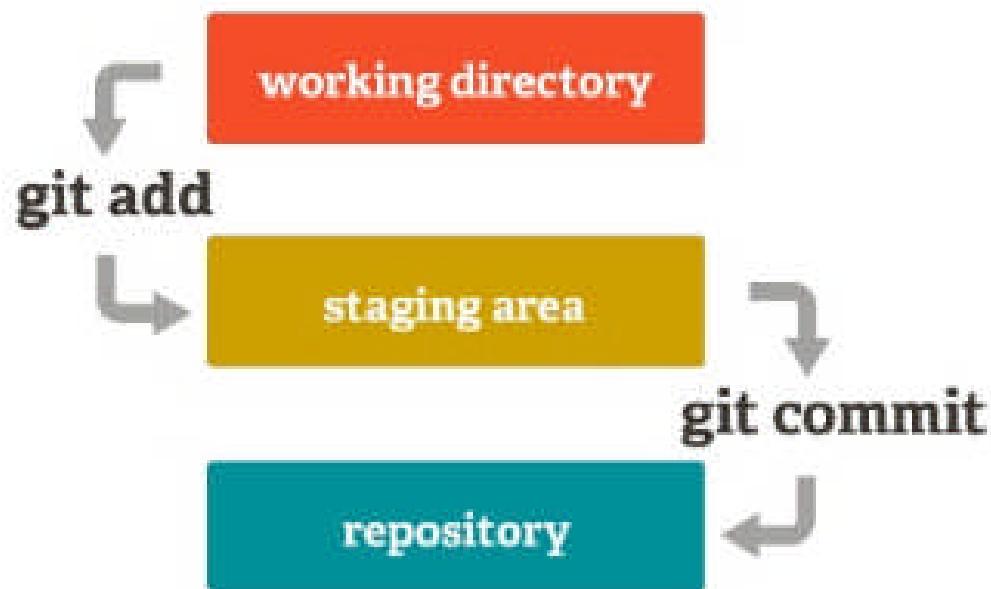
GIT

- Free, Distributed, Fast.
- By Linus for Linux Dev
- Multiple Repos (push/pull)
- Instant lightweight branching
- Patch (changelist) based,
- - Not file based.

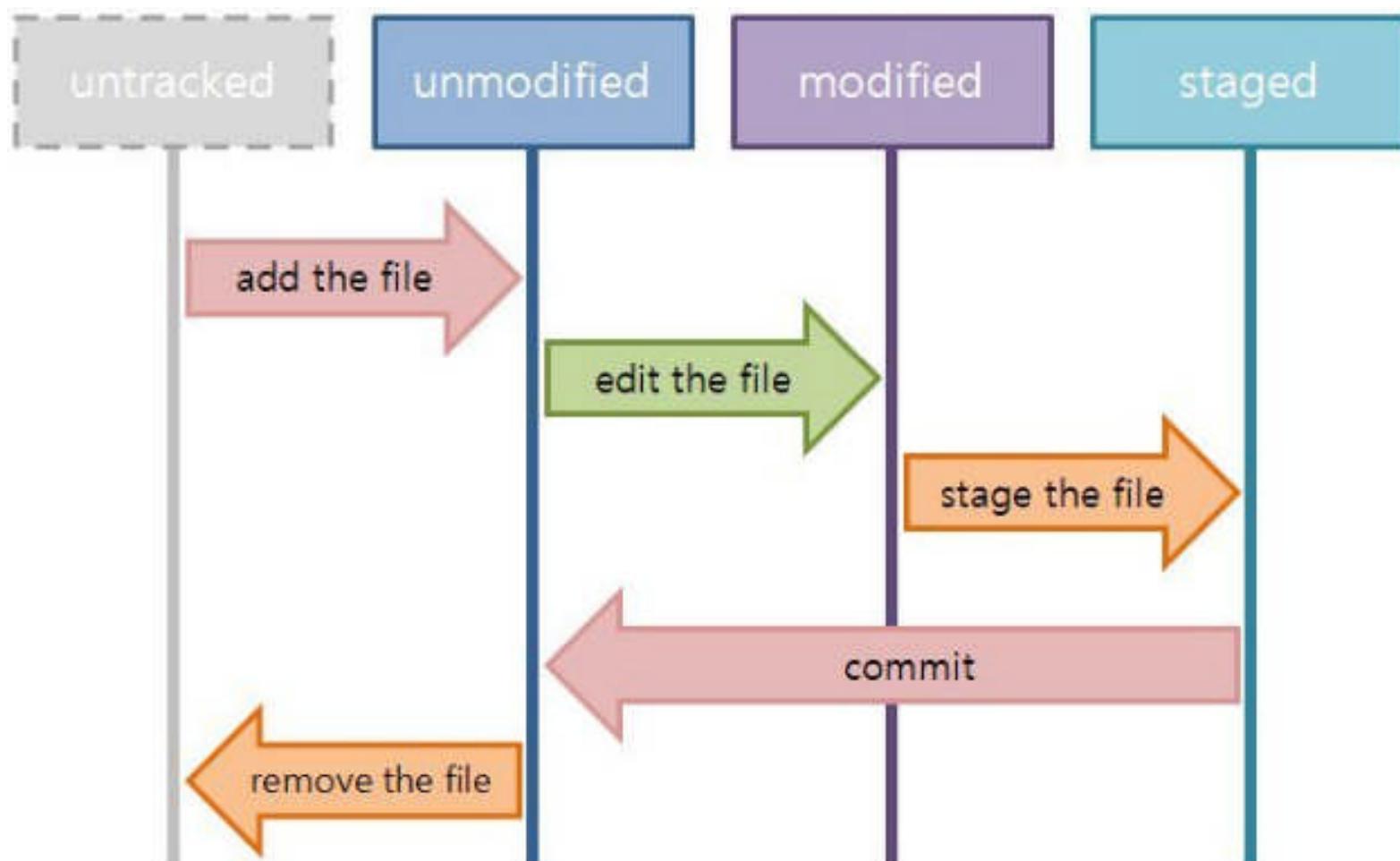


Staging

Git has a staging area (or index), which is an additional layer between the git database and the working copy



File states in Git



Git keeps track of changes

Download and install git if needed

Windows get gitk (with gui).

```
> cd ~/src
```

```
> git init . # Make a local repo in ./git
```

```
> git add . #
```

```
> vi .gitignore # specify files to not put in git
```

```
> vi main.c
```

```
> git commit -m "comment"
```

Remote Repo

```
# Make another Repo /tmp/project.git  
git clone --bare ~/src/.git /tmp/project.git
```

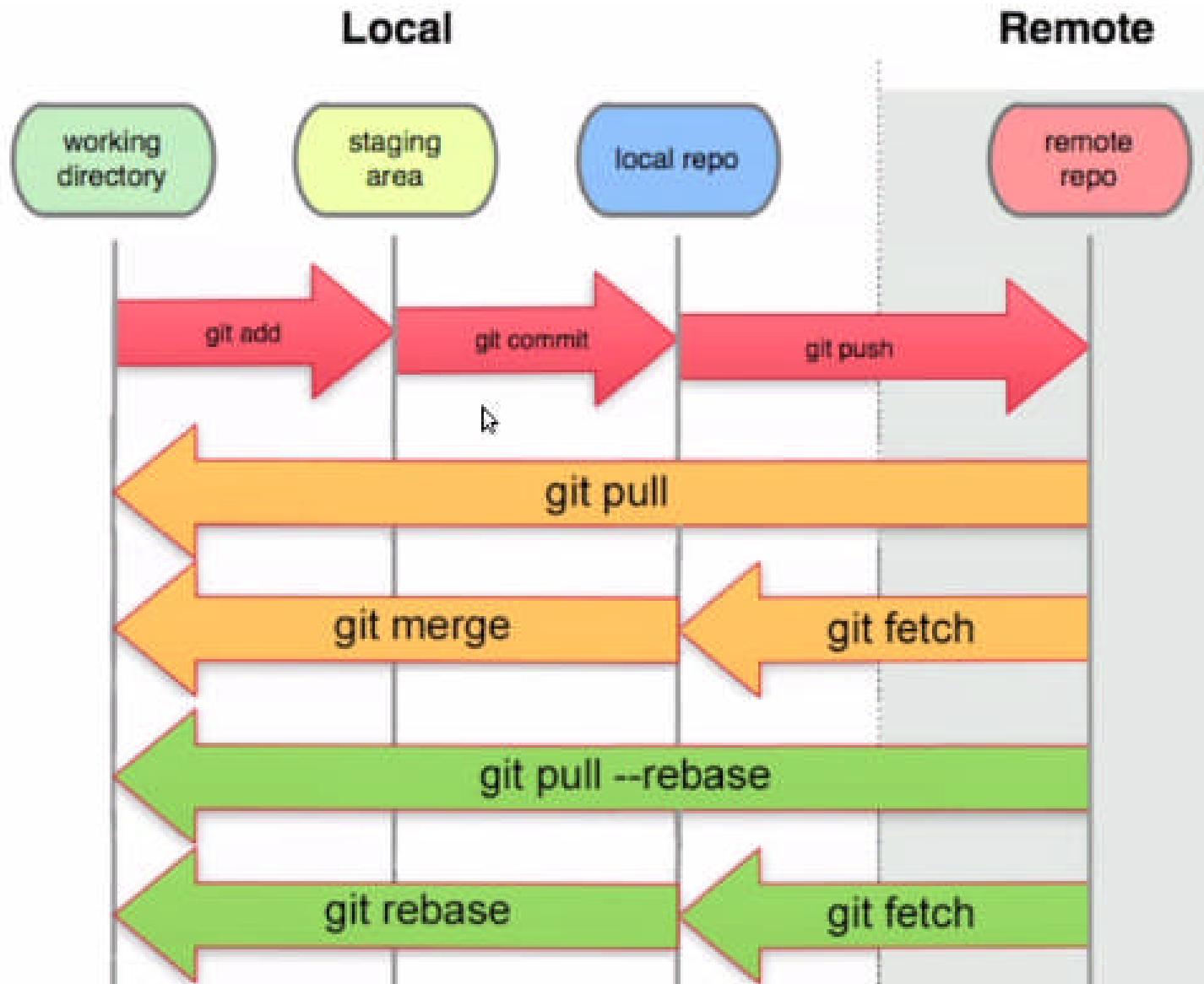
```
# Copy your repo to your server  
scp -r /tmp/project.git ssh://example.com/~/www/
```

Push and Pull to sync repos

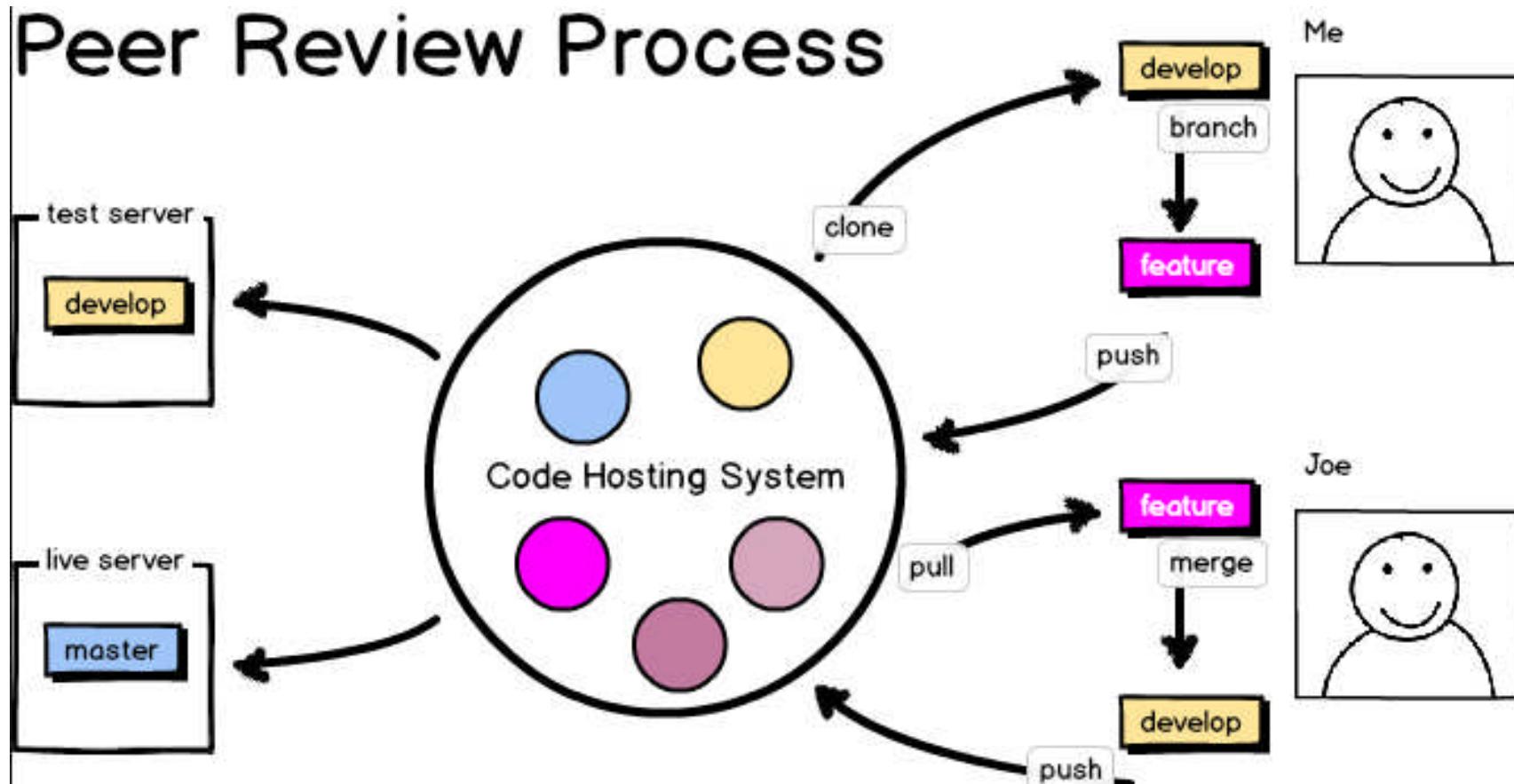
```
> git push ssh://example.com/~/www/project.git
```

```
> git pull http://git.example.com/project.git
```

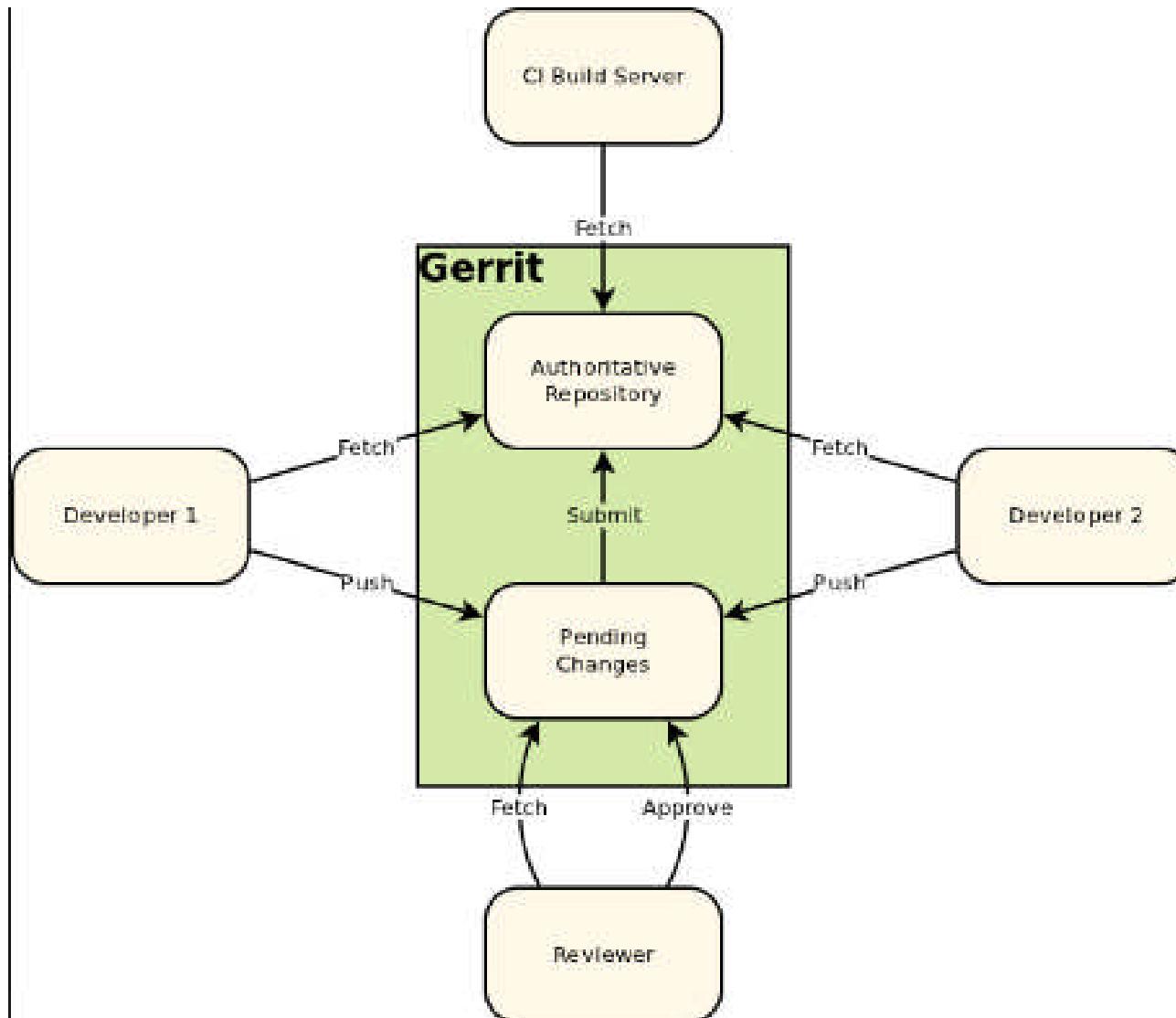
Local and Remote Repos



Team Dev



code reviews: LGTM



Stash - temp saving

```
> git stash # Keep your changes aside
```

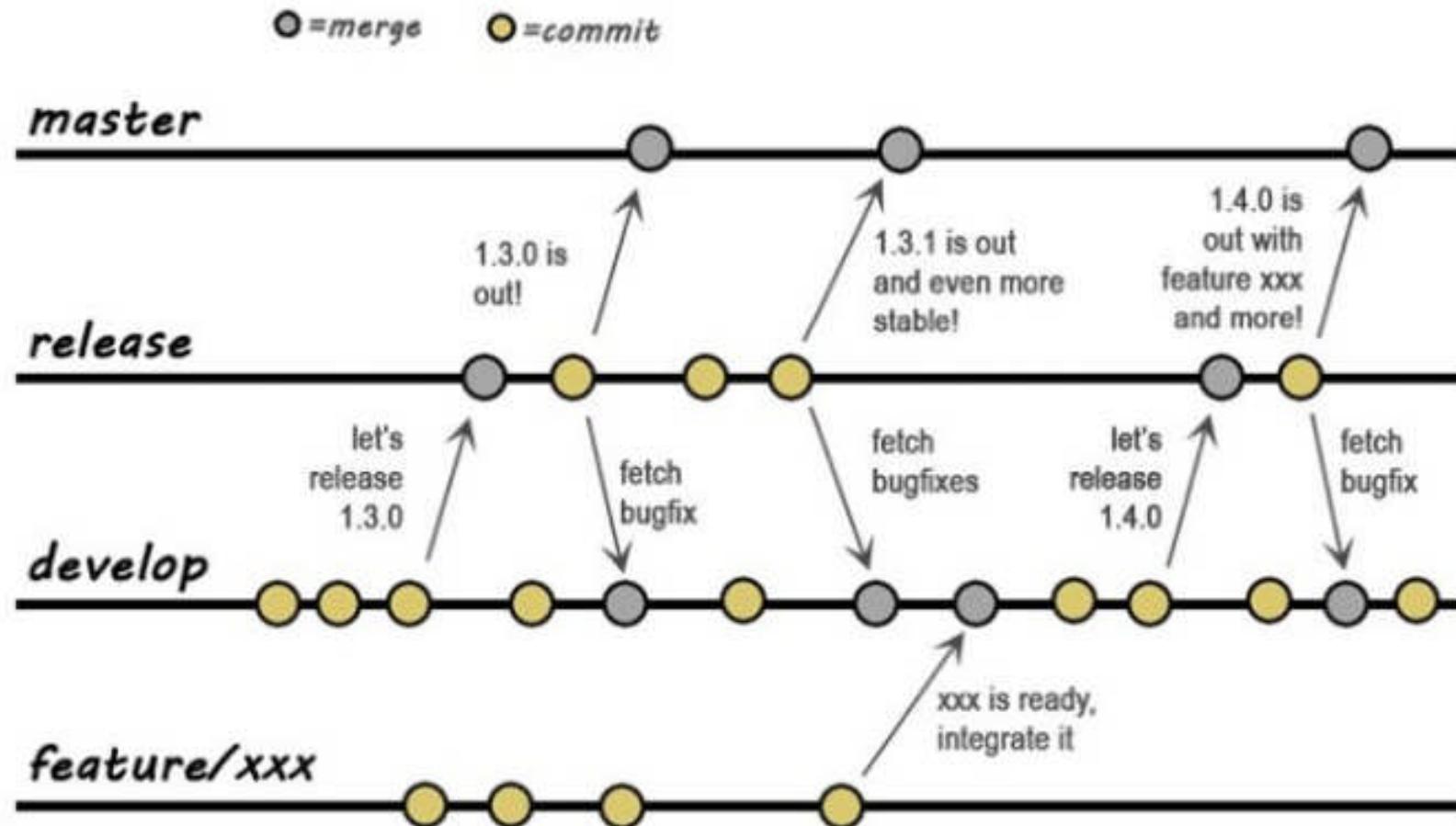
Do some work in other branches

```
> git stash apply # bring back your edits
```

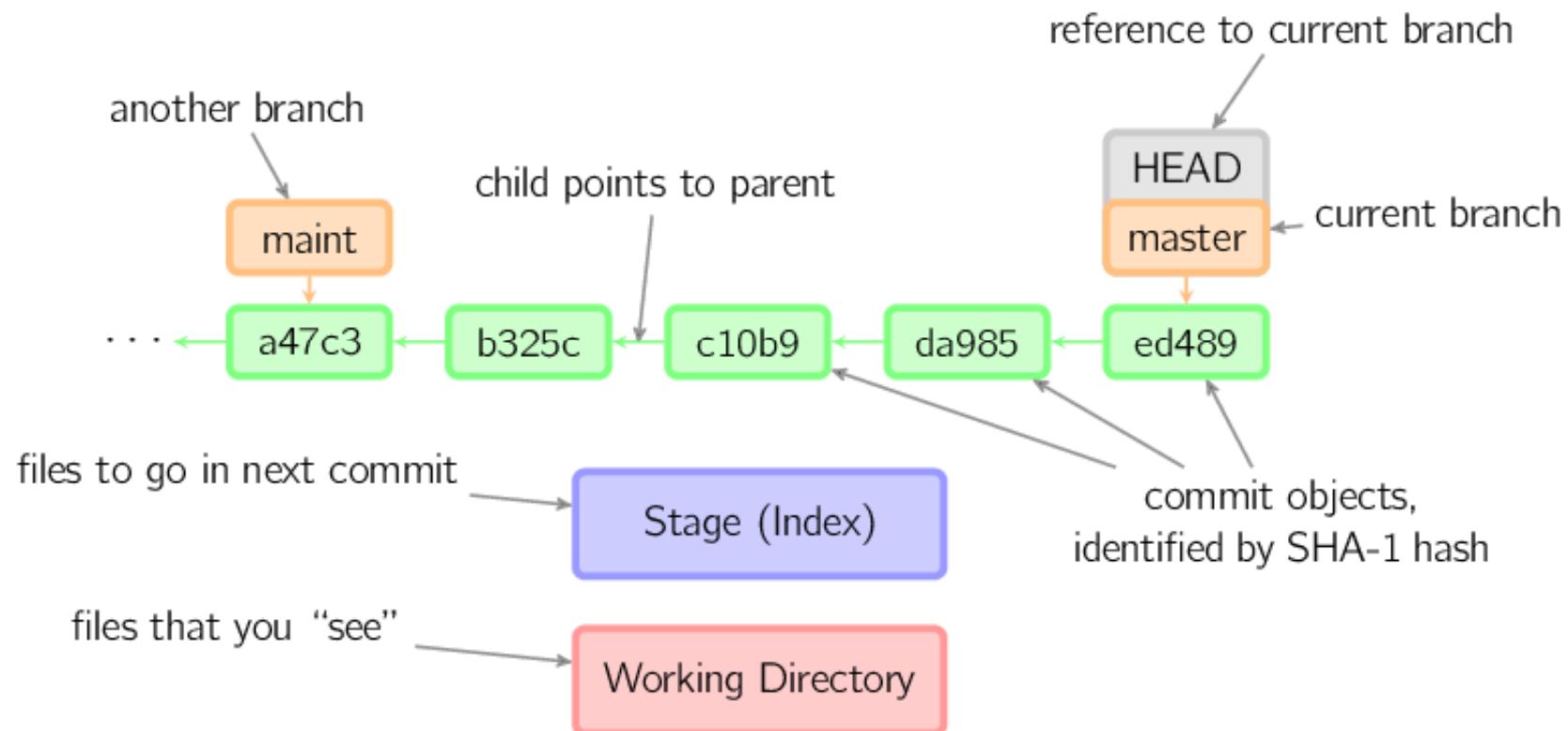
Branch

```
> git branch test      # make a temp branch  
> git checkout test  # Use it  
Do you testing here  
> git diff            # see what changed  
> git diff master...test # compare 2 branches  
> git checkout master # switch branch  
> git merge test      # Keep the good code  
> git branch -d test # delete test branch
```

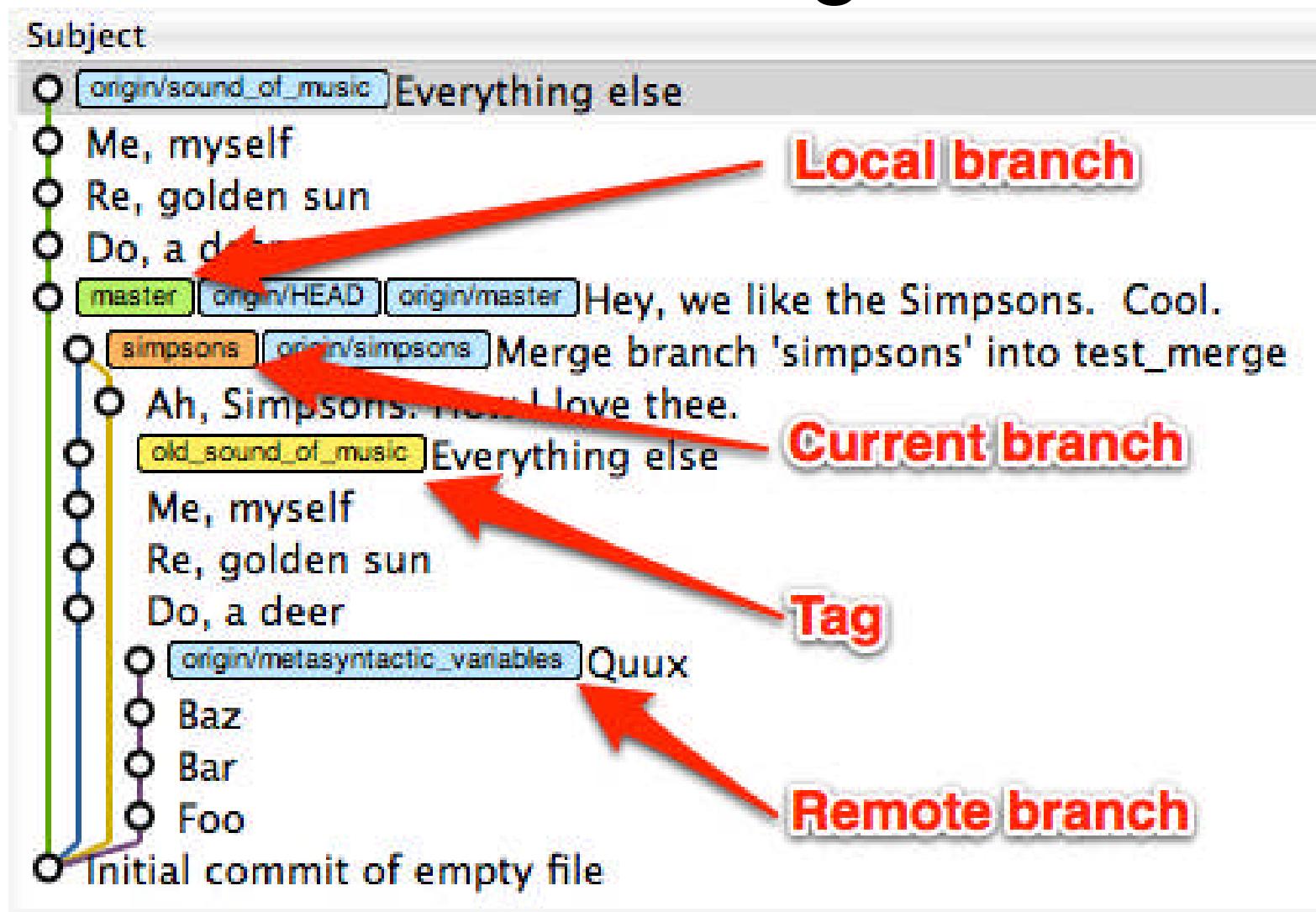
Multiple Branches



SHA1 hash of commit objects

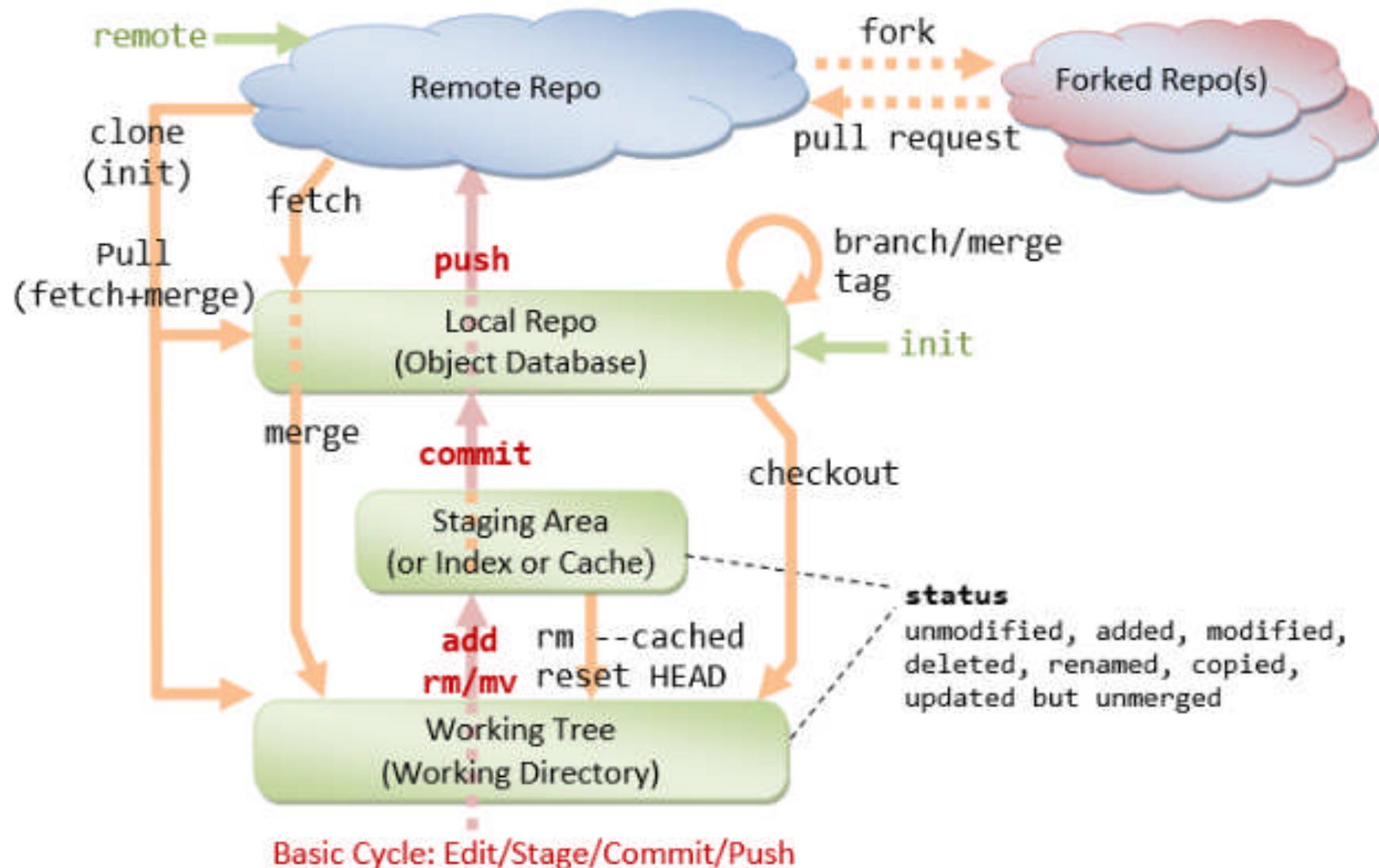


Git log



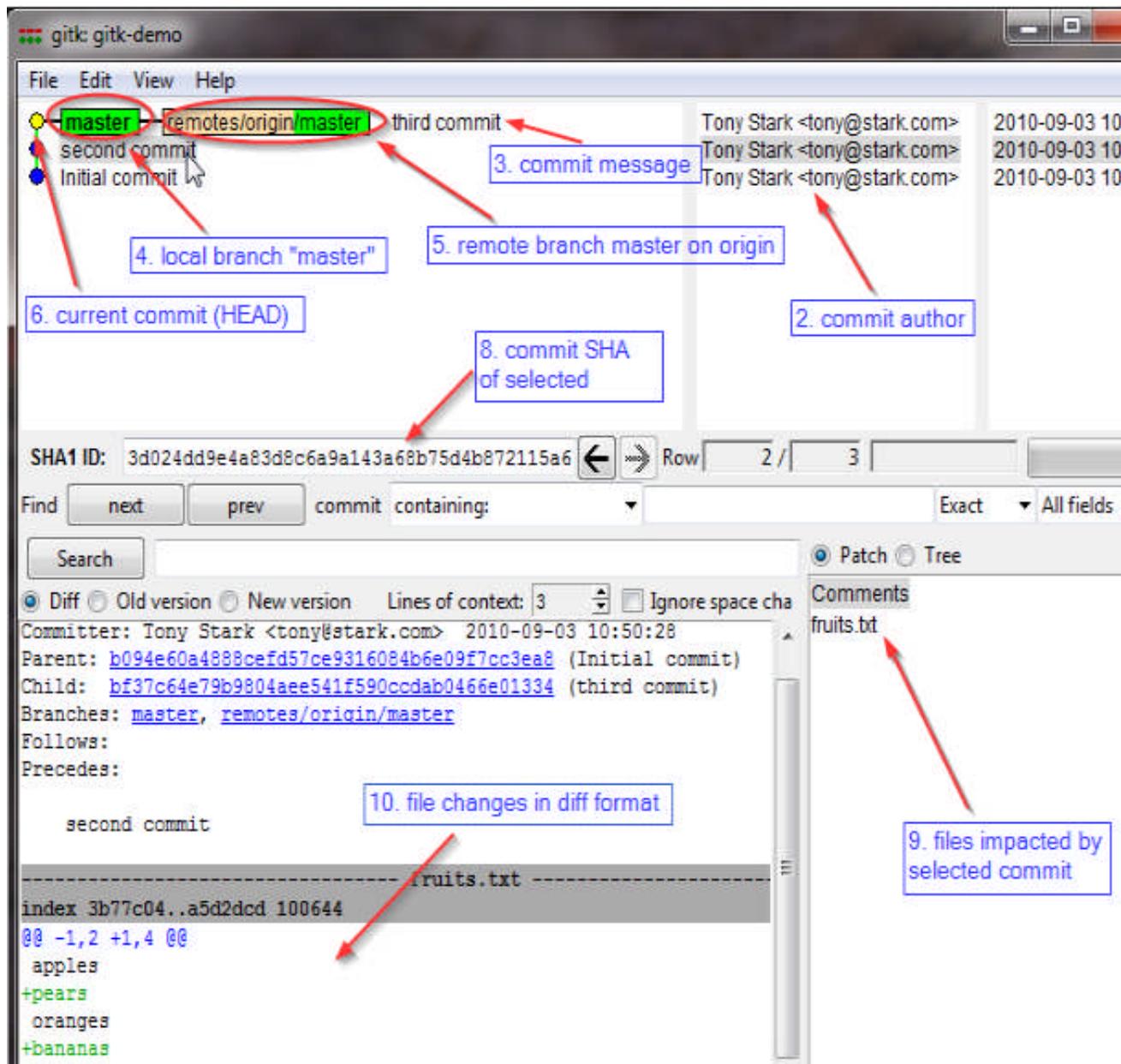
```
> git log --oneline --abbrev-commit --all --graph --decorate
```

Git Dataflow

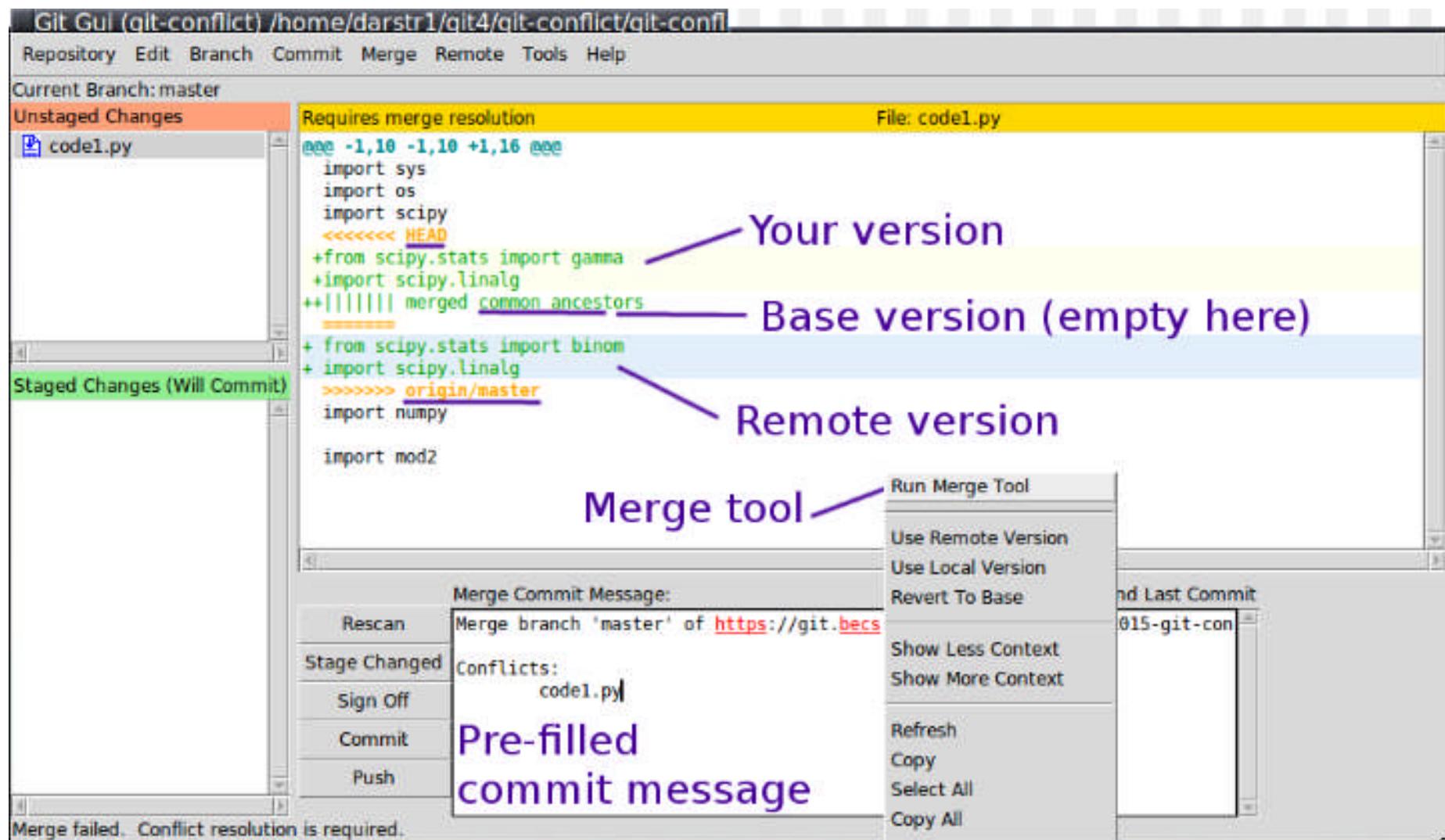


Gitk

- Viewing changelists



Git GUI for commit



Pre-filled
commit message

tig - terminal git history browser

```
fonseca@antimatter: /home/fonseca/src/git/git
2006-05-09 19:32 Junio C Hamano      [master] [next] Merge branch 'master' into next
2006-05-09 19:24 Junio C Hamano      - Merge branch 'fix'
2006-05-09 19:23 Junio C Hamano      - checkout: use --aggressive when running a 3-way merge (-m).
2006-05-09 19:22 Linus Torvalds     - revert/cherry-pick: use aggressive merge.
2006-05-09 16:52 Junio C Hamano     - Merge branch 'jc/clean'
2006-05-09 16:45 Junio C Hamano     - Merge branch 'mw/alternates'
2006-05-09 16:44 Junio C Hamano     - Merge branch 'jc/xshal'
2006-05-09 16:40 Junio C Hamano     - Merge branch 'jc/again'
2006-05-09 16:40 Junio C Hamano     - Merge branch 'np/delta'
2006-05-09 14:16 Junio C Hamano     - Merge branch 'jc/bindiff'

[main] 8d7a397aab561d3782f531e733b617e0e211f04a - commit 3 of 577 (0%)
commit 8d7a397aab561d3782f531e733b617e0e211f04a
Author: Junio C Hamano <junkio@cox.net>
Date:   Tue May  9 19:23:23 2006 -0700

    checkout: use --aggressive when running a 3-way merge (-m).

    After doing an in-index 3-way merge, we always do the stock
    "merge-index merge-one-file" without doing anything fancy;
    use of --aggressive helps performance quite a bit.

    Signed-off-by: Junio C Hamano <junkio@cox.net>
---
git-checkout.sh |    2 ++
1 files changed, 1 insertions(+), 1 deletions(-)

diff --git a/git-checkout.sh b/git-checkout.sh
index 463ed2e..a11c939 100755
--- a/git-checkout.sh
+++ b/git-checkout.sh
[diff] 8d7a397aab561d3782f531e733b617e0e211f04a - line 1 of 28 (3%)
Loaded 28 lines in 0 seconds
```

Vim has fugitive.vim :GV (git history browser inside vim)

```
jez Use only one key[event] handler for all modes.
jez Cosmetics
jez Deactivate hintKeystrokeQueue.
jez Cosmetics
jez Add a namespace for utility linkHints functions.
jez deactivateMode should always invoke the callback passed to it.
jez Fix matchHintsByKey's return values
jez Begin switch from inheritance to composition
jez Reduce coupling.
jez Make deactivateMode() within linkHintsBase async
jez Simplify initializeLinkHints.
jez Adjust line width and other formatting.
jez Remove 'Key Down' log message.
jez Reduce nesting in getVisibleClickableElements.
jez In filter-mode, 'Enter' should activate the lowest-numbered visible hint.
jez Fix filter hinting for images, and added corresponding test.
Mon Jan 31 15:08:17 2011 +0800
@@ -351,7 +351,7 @@ var alphabetHints = {

    setMarkerAttributes: function(marker, linkHintNumber) {
        var hintText = this.numberToHintString(linkHintNumber, this.digitsNeeded);
-       marker.innerHTML = spanWrap(hintText);
+       marker.innerHTML = hintUtils.spanWrap(hintText);
        marker.setAttribute("hintString", hintText);
        return marker;
    },
@@ -455,7 +455,7 @@ var filterHints = {
    }
    linkText = linkText.trim().toLowerCase();
    marker.setAttribute("hintString", hintText);
-   marker.innerHTML = spanWrap(hintText + (showLinkText ? ":" + linkText : ""));
+   marker.innerHTML = hintUtils.spanWrap(hintText + (showLinkText ? ":" + linkText : ""));
    marker.setAttribute("linkText", linkText);
```

Free Online Repositories

- github
- bitbucket

References

1. GIT <https://git-scm.com/>
2. <http://classic.scottr.org/presentations/git-in-5-minutes/>
3. <http://newtfire.org/dh/explainGitShell.html>
4. <https://lostechies.com/joshuaflanagan/2010/09/03/use-gitk-to-understand-git/>
5. CVS <http://www.fisica.uniud.it/~glast/sw/cvs/cvsiintro.html>

Introduction to C

C Programming Language

Introduction

Fred Kuhns

fredk@cse.wustl.edu

Applied Research Laboratory,
Department of Computer Science and Engineering,
Washington University in St. Louis



Introduction to C

- The C programming language was designed by Dennis Ritchie at Bell Laboratories in the early 1970s
- Influenced by
 - ALGOL 60 (1960),
 - CPL (Cambridge, 1963),
 - BCPL (Martin Richard, 1967),
 - B (Ken Thompson, 1970)
- Traditionally used for systems programming.
- Traditional C:
 - *The C Programming Language*, by Brian Kernighan and Dennis Ritchie, 2nd Edition, Prentice Hall
 - Referred to as K&R

Standard C

- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C
- International standard (ISO) in 1990 which was adopted by ANSI and is known as *C89*
- As part of the normal evolution process the standard was updated in 1995 (*C95*) and 1999 (*C99*)
- *C++* and *C*
 - *C++* extends *C* to include support for Object Oriented Programming and other features that facilitate large software development projects
 - *C* is not strictly a subset of *C++*, but it is possible to write "*Clean C*" that conforms to both the *C++* and *C* standards.

Elements of a C Program

- A C development environment includes
 - *System libraries and headers*: a set of standard libraries and their header files. For example see /usr/include and glibc.
 - *Application Source*: application source and header files
 - *Compiler*: converts source to object code for a specific platform
 - *Linker*: resolves external references and produces the executable module
- User program structure
 - there must be one main function where execution begins when the program is run. This function is called main
 - int main (void) { ... },
 - int main (int argc, char *argv[]) { ... }
 - UNIX Systems have a 3rd way to define main(), though it is not POSIX.1 compliant
int main (int argc, char *argv[], char *envp[])
 - additional local and external functions and variables

A Simple C Program

- *Create example file:* try.c
- *Compile using gcc:*
gcc -o try try.c
- The standard C library *libc* is included automatically
- *Execute program*
.try
- Normal termination:
void **exit**(int status);
 - calls functions registered with atexit()
 - flush output streams
 - close all open streams
 - return status value and control to host environment

```
/* you generally want to
 * include stdio.h and
 * stdlib.h
 * */
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
    printf("Hello World\n");
    exit(0);
}
```

Source and Header files

- Just as in C++, place related code within the same module (i.e. file).
- Header files (*.h) export interface definitions
 - function prototypes, data types, macros, inline functions and other common declarations
- Do not place source code (i.e. definitions) in the header file with a few exceptions.
 - inline'd code
 - class definitions
 - const definitions
- C preprocessor (cpp) is used to insert common definitions into source files
- There are other cool things you can do with the preprocessor

Another Example C Program

/usr/include/stdio.h

```
/* comments */  
#ifndef _STDIO_H  
#define _STDIO_H  
  
... definitions and prototypes  
  
#endif
```

/usr/include/stdlib.h

```
/* prevents including file  
 * contents multiple  
 * times */  
#ifndef _STDLIB_H  
#define _STDLIB_H  
  
... definitions and prototypes  
  
#endif
```

#include directs the preprocessor to "include" the contents of the file at this point in the source file.
#define directs preprocessor to define macros.

example.c

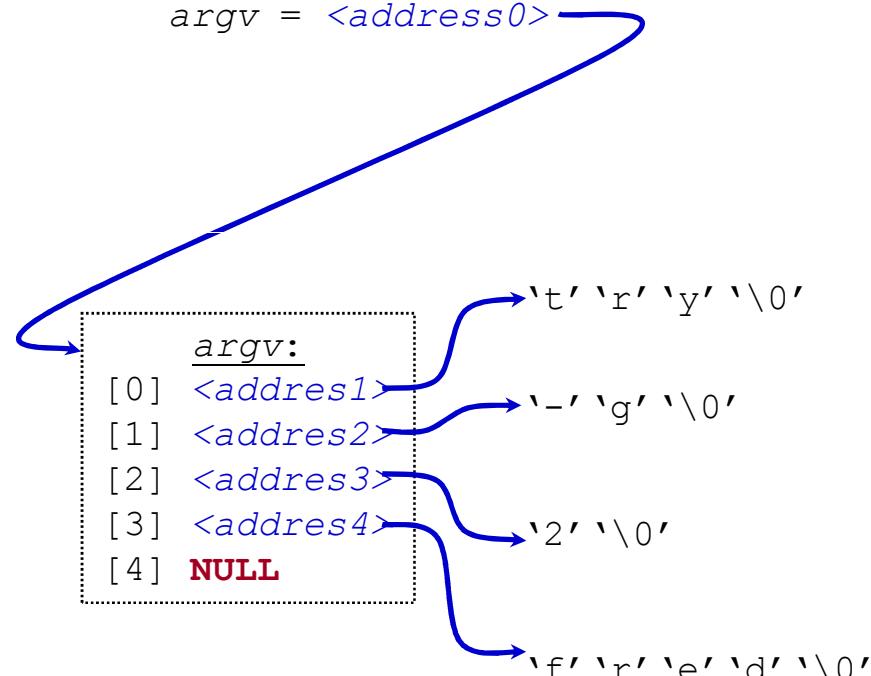
```
/* This is a C-style comment.  
 * You generally want to place  
 * all file includes at start of file  
 */  
#include <stdio.h>  
#include <stdlib.h>  
  
int  
main (int argc, char **argv)  
{  
    // this is a C++-style comment  
    // printf prototype in stdio.h  
    printf("Hello, Prog name = %s\n",  
          argv[0]);  
    exit(0);  
}
```

Passing Command Line Arguments

- When you execute a program you can include arguments on the command line.
- The run time environment will create an argument vector.
 - argv is the argument vector
 - argc is the number of arguments
- Argument vector is an array of pointers to strings.
- a *string* is an array of characters terminated by a binary 0 (NULL or '\0').
- argv[0] is always the program name, so argc is at least 1.

```
./try -g 2 fred
```

argc = 4,
argv = <address0>



C Standard Header Files you may want to use

- Standard Headers you should know about:
 - stdio.h - file and console (also a file) IO: *perror, printf, open, close, read, write, scanf, etc.*
 - stdlib.h - common utility functions: *malloc, calloc, strtol, atoi, etc*
 - string.h - string and byte manipulation: *strlen, strcpy, strcat, memcpy, memset, etc.*
 - ctype.h - character types: *isalnum, isprint, isupper, tolower, etc.*
 - errno.h - defines *errno* used for reporting system errors
 - math.h - math functions: *ceil, exp, floor, sqrt, etc*
 - signal.h - signal handling facility: *raise, signal, etc*
 - stdint.h - standard integer: *intN_t, uintN_t, etc*
 - time.h - time related facility: *asctime, clock, time_t, etc.*

The Preprocessor

- The C preprocessor permits you to define simple macros that are evaluated and expanded prior to compilation.
- Commands begin with a '#'. Abbreviated list:
 - `#define` : defines a macro
 - `#undef` : removes a macro definition
 - `#include` : insert text from file
 - `#if` : conditional based on value of expression
 - `#ifdef` : conditional based on whether macro defined
 - `#ifndef` : conditional based on whether macro is not defined
 - `#else` : alternative
 - `#elif` : conditional alternative
 - `defined()` : preprocessor function: 1 if name defined, else 0
 - `#if defined(__NetBSD__)`

Preprocessor: Macros

- Using macros as functions, exercise caution:
 - flawed example: `#define mymult(a,b) a*b`
 - Source: `k = mymult(i-1, j+5);`
 - Post preprocessing: `k = i - 1 * j + 5;`
 - better: `#define mymult(a,b) (a) * (b)`
 - Source: `k = mymult(i-1, j+5);`
 - Post preprocessing: `k = (i - 1) * (j + 5);`
- Be careful of *side effects*, for example what if we did the following
 - Macro: `#define mysq(a) (a) * (a)`
 - flawed usage:
 - Source: `k = mysq(i++)`
 - Post preprocessing: `k = (i++) * (i++)`
- Alternative is to use inline'ed functions
 - `inline int mysq(int a) {return a*a};`
 - `mysq(i++)` works as expected in this case.

Preprocessor: Conditional Compilation

- Its generally better to use inline'ed functions
- Typically you will use the preprocessor to define constants, perform conditional code inclusion, include header files or to create shortcuts
- ```
#define DEFAULT_SAMPLES 100
```
- ```
#ifdef __linux
    static inline int64_t
       _gettime(void) {...}
```
- ```
#elif defined(sun)
 static inline int64_t
 _gettime(void) {return (int64_t)gethrtime();}
```
- ```
#else
    static inline int64_t
       _gettime(void) {... gettimeofday() ...}
```
- ```
#endif
```

# Another Simple C Program

```
int main (int argc, char **argv) {
 int i;
 printf("There are %d arguments\n", argc);
 for (i = 0; i < argc; i++)
 printf("Arg %d = %s\n", i, argv[i]);
 return 0;
}
```

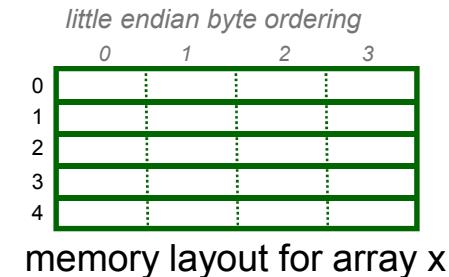
- Notice that the syntax is similar to Java
- What's new in the above simple program?
  - of course you will have to learn the new interfaces and utility functions defined by the C standard and UNIX
  - Pointers will give you the most trouble

# Arrays and Pointers

- A variable declared as an array represents a contiguous region of memory in which the array elements are stored.

```
int x[5]; // an array of 5 4-byte ints.
```

- All arrays begin with an index of 0



- An array identifier is equivalent to a pointer that references the first element of the array

- int x[5], \*ptr;  
ptr = &x[0] **is equivalent to** ptr = x;

- Pointer arithmetic and arrays:

- int x[5];  
x[2] **is the same as** \* (x + 2), the compiler will assume you mean 2 objects beyond element x.

# Pointers

- For any type T, you may form a pointer type to T.
  - Pointers may reference a function or an object.
  - The value of a pointer is the address of the corresponding object or function
  - Examples: `int *i; char *x; int (*myfunc)();`
- Pointer operators: `*` dereferences a pointer, `&` creates a pointer (reference to)
  - `int i = 3; int *j = &i;  
*j = 4; printf("i = %d\n", i); // prints i = 4`
  - `int myfunc (int arg);  
int (*fptr)(int) = myfunc;  
i = fptr(4); // same as calling myfunc(4);`
- Generic pointers:
  - Traditional C used (`char *`)
  - Standard C uses (`void *`) - these can not be dereferenced or used in pointer arithmetic. So they help to reduce programming errors
- Null pointers: use `NULL` or `0`.
- *It is a good idea to always initialize pointers to NULL.*

# Pointers in C (and C++)

Step 1:

```
int main (int argc, argv) {
 int x = 4;
 int *y = &x;
 int *z[4] = {NULL, NULL, NULL, NULL};
 int a[4] = {1, 2, 3, 4};
 ...
```

Note: The compiler converts z[1] or \*(z+1) to  
*Value at address (Address of z + sizeof(int));*

In C you would write the byte address as:

```
(char *)z + sizeof(int);
```

or letting the compiler do the work for you

```
(int *)z + 1;
```

| Program Memory | Address |
|----------------|---------|
| x              | 4       |
| y              | 0x3dc   |
|                | 0x3d8   |
|                | NA      |
|                | 0x3d4   |
|                | NA      |
|                | 0x3d0   |
| z[3]           | 0       |
| z[2]           | 0       |
| z[1]           | 0       |
| z[0]           | 0       |
|                | 0x3cc   |
|                | 0x3c8   |
|                | 0x3c4   |
|                | 0x3c0   |
| a[3]           | 4       |
| a[2]           | 3       |
| a[1]           | 2       |
| a[0]           | 1       |
|                | 0x3bc   |
|                | 0x3b8   |
|                | 0x3b4   |
|                | 0x3b0   |

# Pointers Continued

Step 1:

```
int main (int argc, argv) {
 int x = 4;
 int *y = &x;
 int *z[4] = {NULL, NULL, NULL, NULL};
 int a[4] = {1, 2, 3, 4};
```

Step 2: Assign addresses to array Z

```
z[0] = a; // same as &a[0];
z[1] = a + 1; // same as &a[1];
z[2] = a + 2; // same as &a[2];
z[3] = a + 3; // same as &a[3];
```

| Program Memory | Address |
|----------------|---------|
| x              | 4       |
| y              | 0x3dc   |
|                | NA      |
|                | NA      |
| z[3]           | 0x3bc   |
| z[2]           | 0x3b8   |
| z[1]           | 0x3b4   |
| z[0]           | 0x3b0   |
| a[3]           | 4       |
| a[2]           | 3       |
| a[1]           | 2       |
| a[0]           | 1       |

# Pointers Continued

Step 1:

```
int main (int argc, argv) {
 int x = 4;
 int *y = &x;
 int *z[4] = {NULL, NULL, NULL, NULL};
 int a[4] = {1, 2, 3, 4};
```

Step 2:

```
z[0] = a;
z[1] = a + 1;
z[2] = a + 2;
z[3] = a + 3;
```

Step 3: No change in z's values

```
z[0] = (int *)((char *)a);
z[1] = (int *)((char *)a
 + sizeof(int));
z[2] = (int *)((char *)a
 + 2 * sizeof(int));
z[3] = (int *)((char *)a
 + 3 * sizeof(int));
```

Program Memory Address

|      |       |       |
|------|-------|-------|
|      |       |       |
| x    | 4     | 0x3dc |
| y    | 0x3dc | 0x3d8 |
|      | NA    | 0x3d4 |
|      | NA    | 0x3d0 |
| z[3] | 0x3bc | 0x3cc |
| z[2] | 0x3b8 | 0x3c8 |
| z[1] | 0x3b4 | 0x3c4 |
| z[0] | 0x3b0 | 0x3c0 |
| a[3] | 4     | 0x3bc |
| a[2] | 3     | 0x3b8 |
| a[1] | 2     | 0x3b4 |
| a[0] | 1     | 0x3b0 |
|      |       |       |

# Getting Fancy with Macros

```
#define QNODE(type) \
struct { \
 struct type *next; \
 struct type **prev; \
}

#define QNODE_INIT(node, field) \
do { \
 (node)->field.next = (node); \
 (node)->field.prev = \
 &(node)->field.next; \
} while (/* */ 0);

#define QFIRST(head, field) \
 ((head)->field.next)

#define QNEXT(node, field) \
 ((node)->field.next)

#define QEMPTY(head, field) \
 ((head)->field.next == (head))

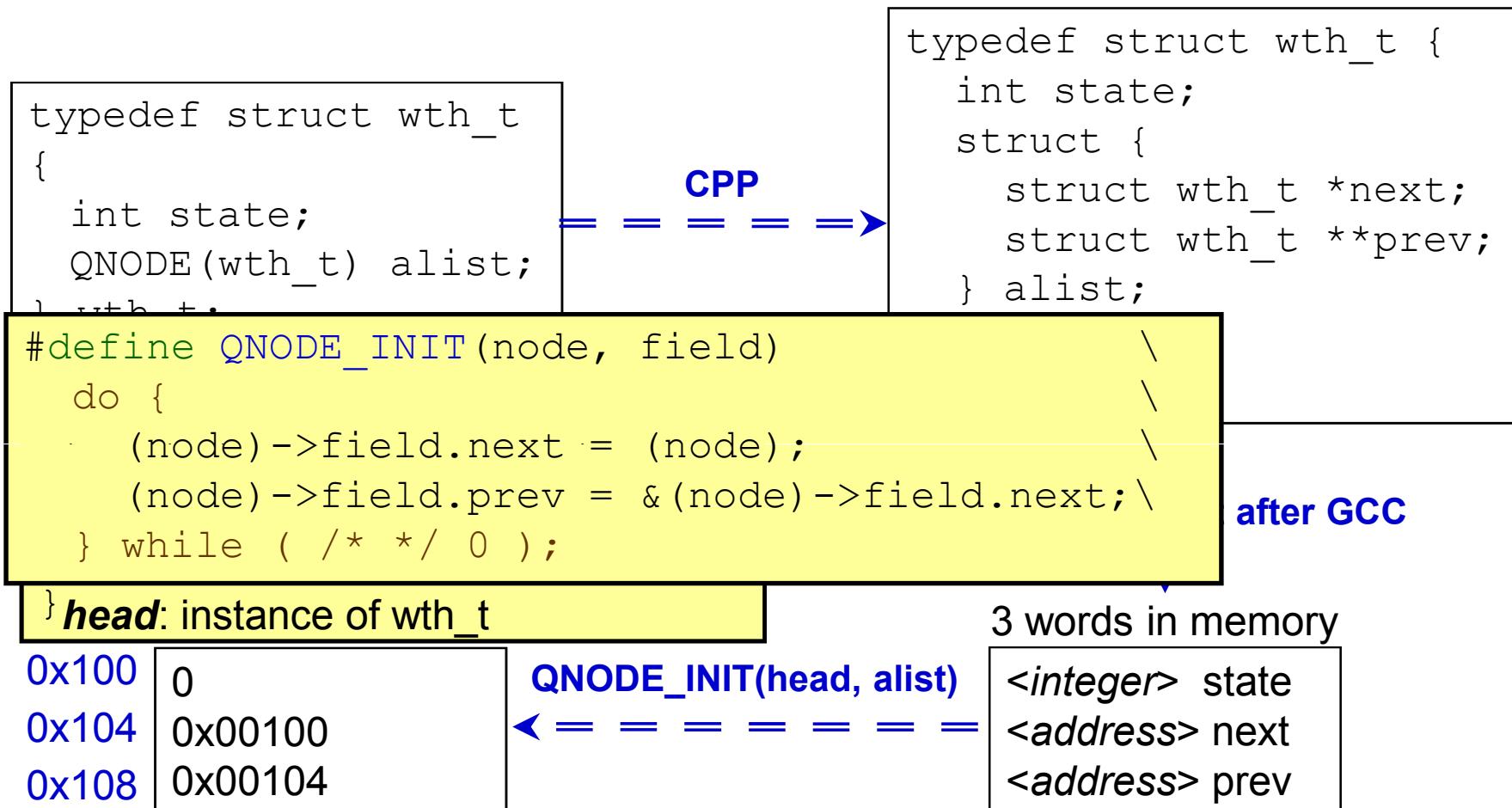
#define QFOREACH(head, var, field) \
for ((var) = (head)->field.next; \
 (var) != (head); \
 (var) = (var)->field.next)

#define QINSERT_BEFORE(loc, node, field) \
do { \
 *(loc)->field.prev = (node); \
 (node)->field.prev = \
 (loc)->field.prev; \
 (loc)->field.prev = \
 &((node)->field.next); \
 (node)->field.next = (loc); \
} while (/* */ 0)

#define QINSERT_AFTER(loc, node, field) \
do { \
 ((loc)->field.next)->field.prev = \
 &(node)->field.next; \
 (node)->field.next = (loc)->field.next; \
 (loc)->field.next = (node); \
 (node)->field.prev = &(loc)->field.next; \
} while (/* */ 0)

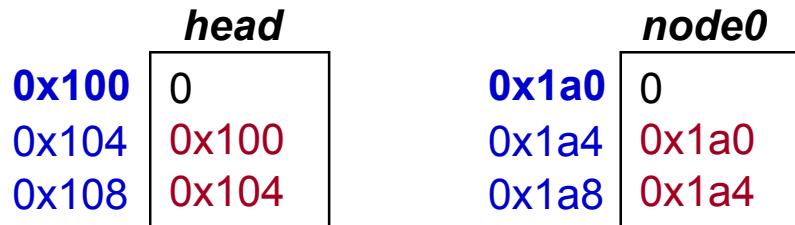
#define QREMOVE(node, field) \
do { \
 *((node)->field.prev) = (node)->field.next; \
 ((node)->field.next)->field.prev = \
 (node)->field.prev; \
 (node)->field.next = (node); \
 (node)->field.prev = &((node)->field.next); \
} while (/* */ 0)
```

# After Preprocessing and Compiling



# QNODE Manipulations

*before*



```
#define QINSERT_BEFORE(head, node, alist)\\
do {\\
 *(head)->alist.prev = (node);\\
 (node)->alist.prev = (head)->alist.prev; \\
 (head)->alist.prev = &(node)->alist.next;\\
 (node)->alist.next = (head);\\
} while /* */0)
```

QINSERT\_BEFORE(head, node0, alist);

?

# QNODE Manipulations

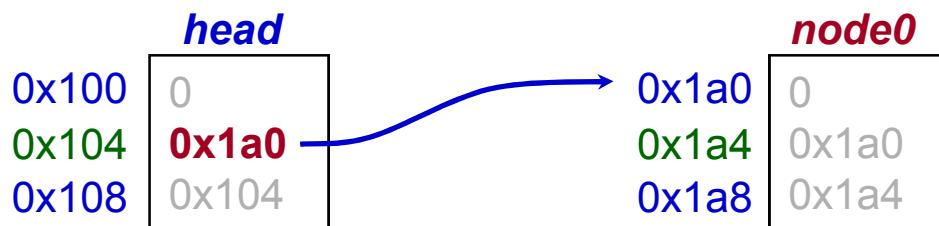
*before*

|       | <i>head</i> |
|-------|-------------|
| 0x100 | 0           |
| 0x104 | 0x100       |
| 0x108 | 0x104       |

|       | <i>node0</i> |
|-------|--------------|
| 0x1a0 | 0            |
| 0x1a4 | 0x1a0        |
| 0x1a8 | 0x1a4        |

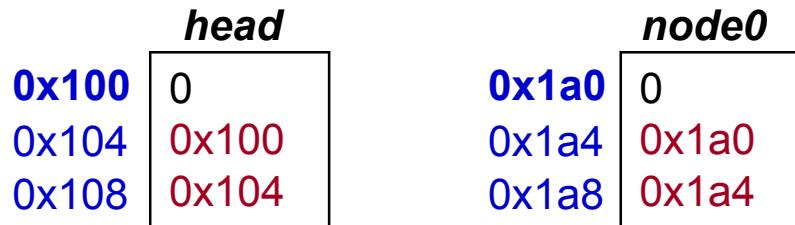
```
#define QINSERT_BEFORE(head, node, alist)\\
 do {\\
 *(head)->alist.prev = (node);\\
 (node)->alist.prev = (head)->alist.prev; \\ \\
 (head)->alist.prev = &(node)->alist.next;\\
 (node)->alist.next = (head);\\
 } while /* */0)
```

QINSERT BEFORE(head, node0, alist);



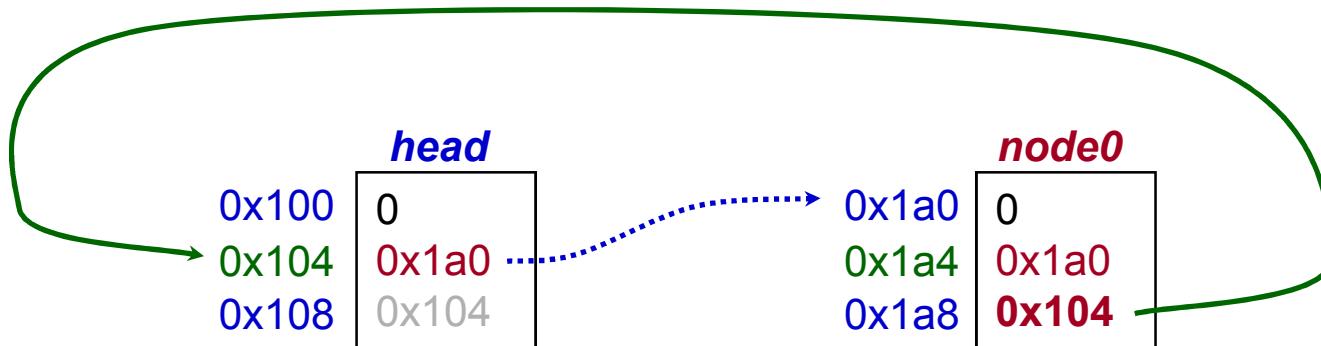
# QNODE Manipulations

*before*



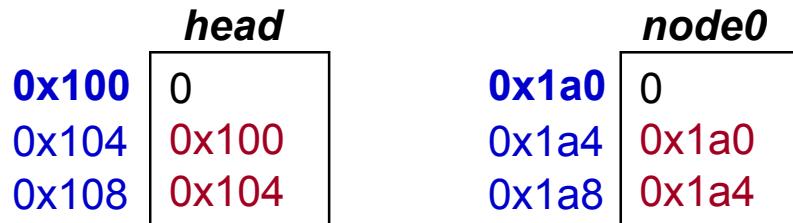
```
#define QINSERT_BEFORE(head, node, alist)\\
do {\\
 *(head)->alist.prev = (node);\\
 (node)->alist.prev = (head)->alist.prev; \\
 (head)->alist.prev = &(node)->alist.next;\\
 (node)->alist.next = (head);\\
} while /* */0)
```

QINSERT\_BEFORE(head, node0, alist);



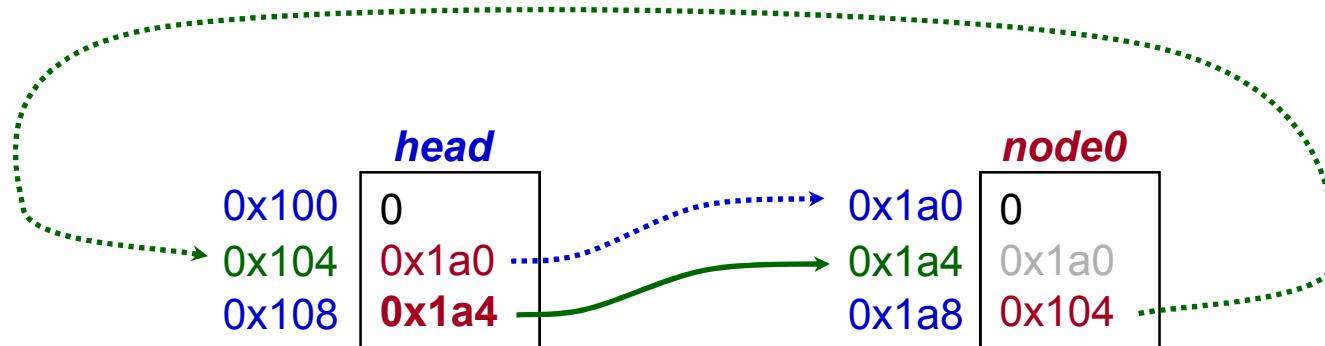
# QNODE Manipulations

*before*



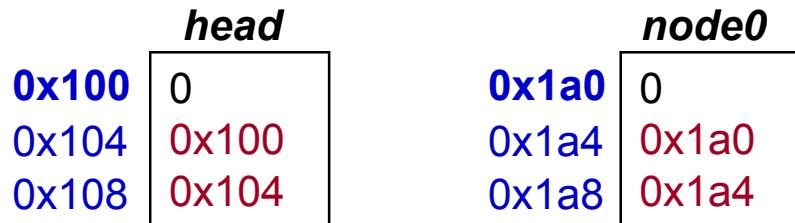
```
#define QINSERT_BEFORE(head, node, alist) \
 do { \
 *(head)->alist.prev = (node); \
 (node)->alist.prev = (head)->alist.prev; \
 (head)->alist.prev = &(node)->alist.next; \
 (node)->alist.next = (head); \
 } while /* */0
```

QINSERT BEFORE(head, node0, alist);



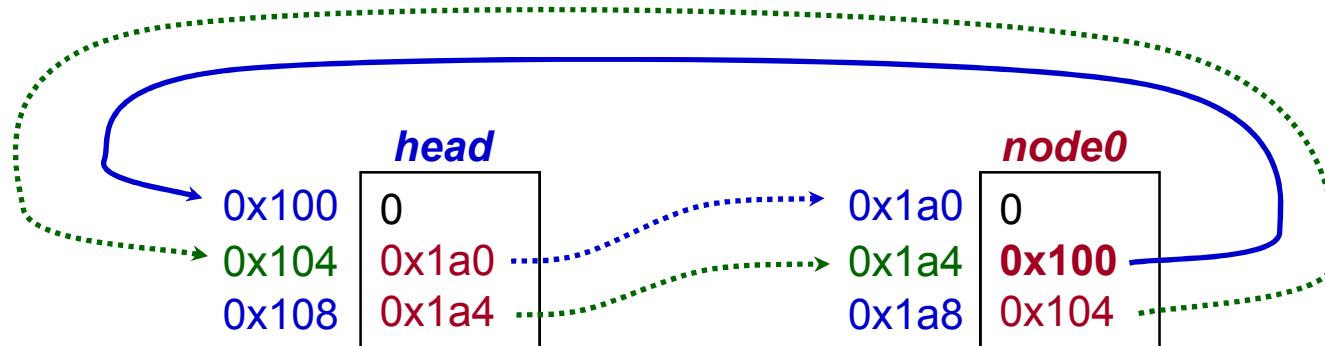
# QNODE Manipulations

*before*



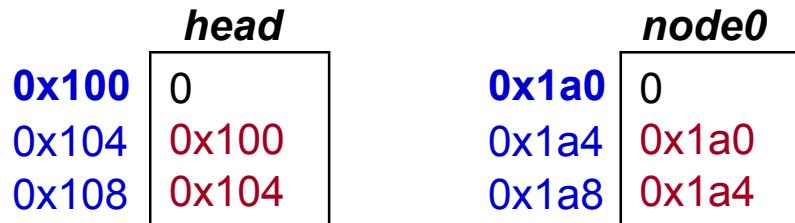
```
#define QINSERT_BEFORE(head, node, alist) \
do { \
 *(head)->alist.prev = (node); \
 (node)->alist.prev = (head)->alist.prev; \
 (head)->alist.prev = &(node)->alist.next; \
 (node)->alist.next = (head); \
} while /* */0)
```

QINSERT BEFORE(head, node0, alist);



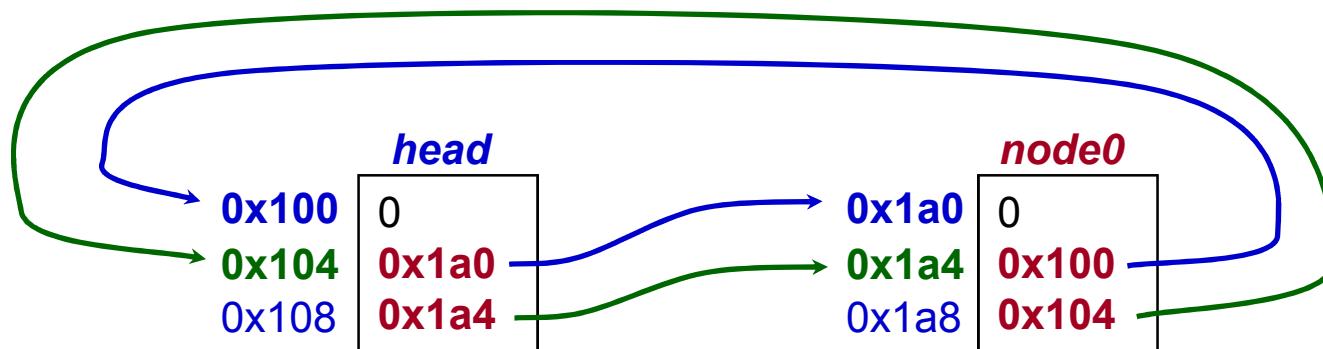
# QNODE Manipulations

*before*



```
#define QINSERT_BEFORE(head, node, alist)\\
do {\\
 *(head)->alist.prev = (node);\\
 (node)->alist.prev = (head)->alist.prev; \\
 (head)->alist.prev = &(node)->alist.next;\\
 (node)->alist.next = (head);\\
} while /* */0)
```

QINSERT BEFORE(head, node0, alist);

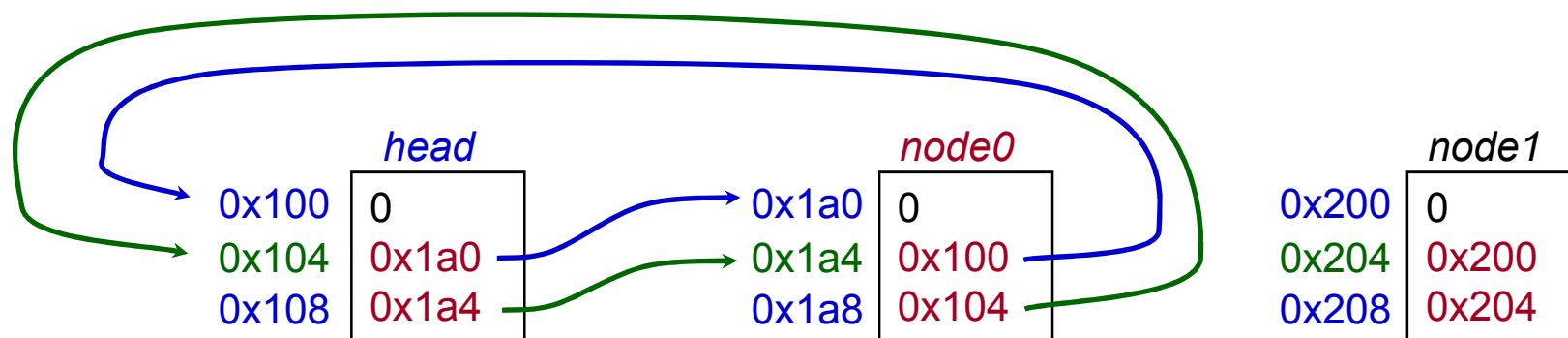


# Adding a Third Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0x1a0        | 0x200        |
| 0x104 | 0x1a0       | 0x1a4        | 0x200        |
| 0x108 | 0x1a4       | 0x100        | 0x204        |

```
#define QINSERT_BEFORE(head, node, alist)\\
do {\\
 *(head)->alist.prev = (node);\\
 (node)->alist.prev = (head)->alist.prev;\\
 (head)->alist.prev = &(node)->alist.next;\\
 (node)->alist.next = (head);\\
} while /* */0)
```

QINSERT\_BEFORE(head, node1, alist);

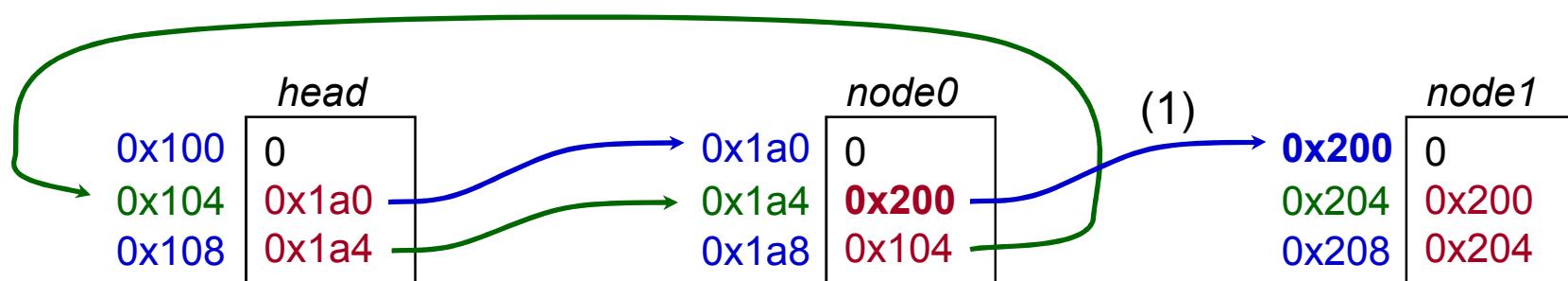


# Adding a Third Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0x1a0        | 0x200        |
| 0x104 | 0x1a0       | 0x1a4        | 0x200        |
| 0x108 | 0x1a4       | 0x1a8        | 0x204        |

```
#define QINSERT_BEFORE(head, node1, alist)\\
 do {\\
 (1) *(head)->alist.prev = (node1);\\
 (node1)->alist.prev = (head)->alist.prev;\\
 (head)->alist.prev = &(node1)->alist.next;\\
 (node1)->alist.next = (head);\\
 } while /* */0)
```

QINSERT\_BEFORE(head, node1, alist);

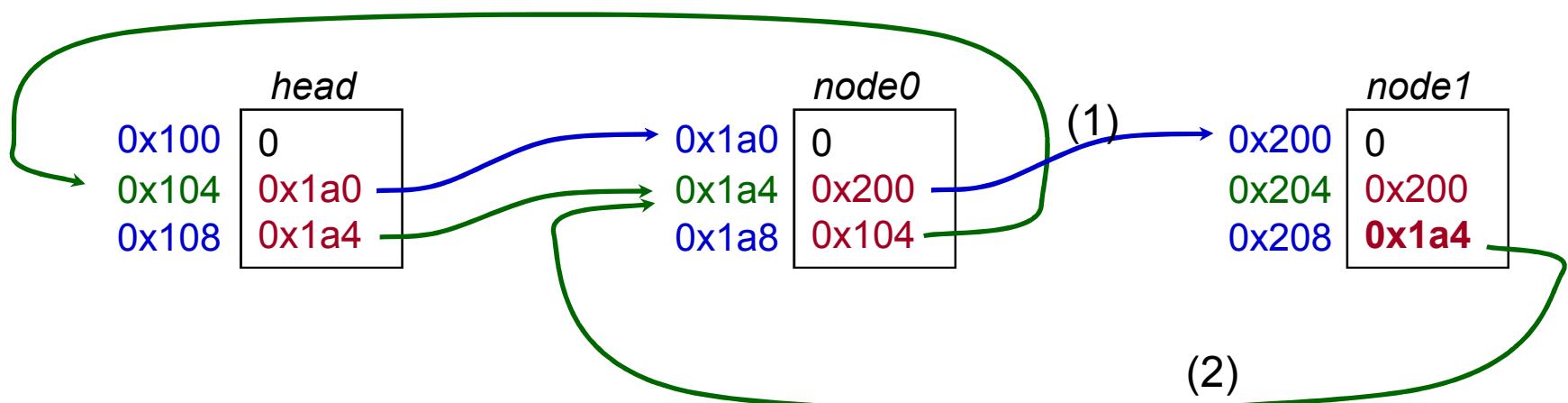


# Adding a Third Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0x1a0        | 0x200        |
| 0x104 | 0x1a0       | 0x1a4        | 0x200        |
| 0x108 | 0x1a4       | 0x100        | 0x204        |

```
#define QINSERT_BEFORE(head, node1, alist)\\
 do {\\
 *(head)->alist.prev = (node1);\\
 (2) (node1)->alist.prev = (head)->alist.prev;\\
 (head)->alist.prev = &(node1)->alist.next;\\
 (node1)->alist.next = (head);\\
 } while /* */0)
```

QINSERT\_BEFORE(head, node1, alist);

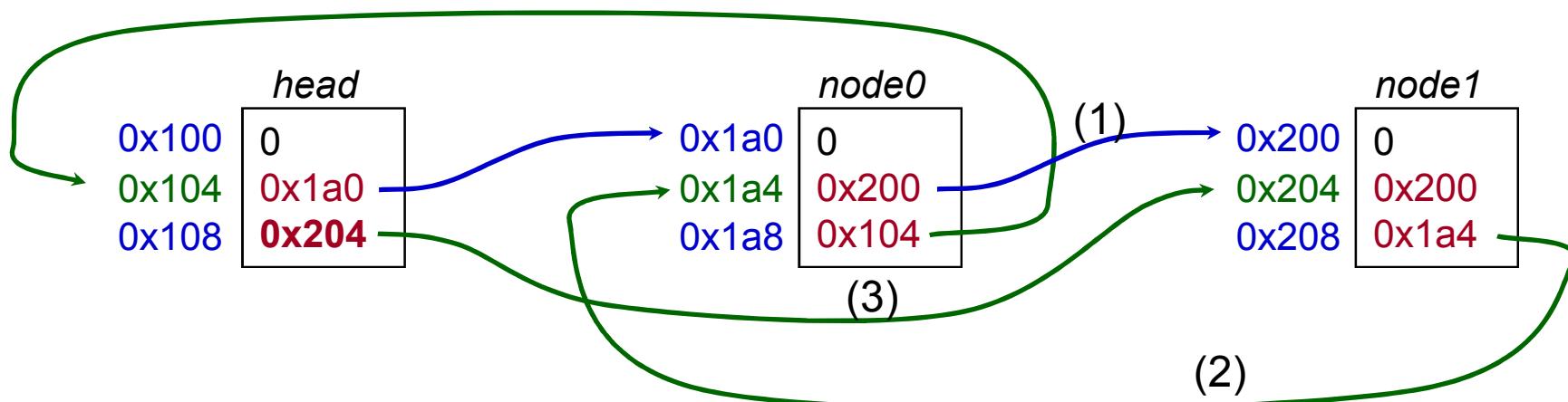


# Adding a Third Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0x1a0        | 0x200        |
| 0x104 | 0x1a0       | 0x1a4        | 0x200        |
| 0x108 | 0x1a4       | 0x100        | 0x204        |

```
#define QINSERT_BEFORE(head, node1, alist)\\
 do {\\
 (1) *(head)->alist.prev = (node1);\\
 (2) (node1)->alist.prev = (head)->alist.prev;\\
 (3) (head)->alist.prev = &(node1)->alist.next;\\
 (node1)->alist.next = (head);\\
 } while /* */0)
```

QINSERT\_BEFORE(head, node1, alist);

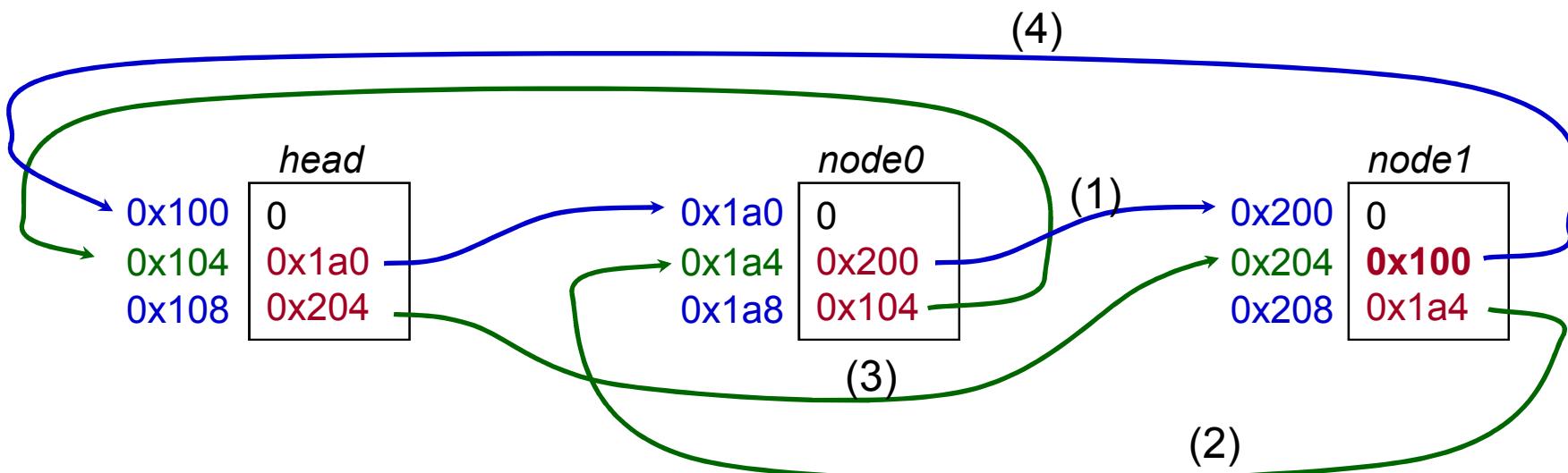


# Adding a Third Node

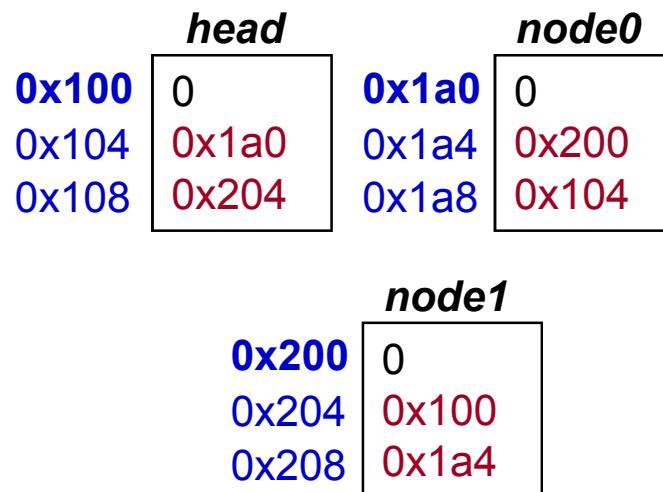
|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0x1a0        | 0x200        |
| 0x104 | 0x1a0       | 0x1a4        | 0x200        |
| 0x108 | 0x1a4       | 0x100        | 0x204        |

```
#define QINSERT_BEFORE(head, node1, alist) \
 do { \
 (1) *(head)->alist.prev = (node1); \
 (2) (node1)->alist.prev = (head)->alist.prev; \
 (3) (head)->alist.prev = &(node1)->alist.next; \
 (4) (node1)->alist.next = (head); \
 } while /* */0)
```

QINSERT BEFORE(head, node1, alist);

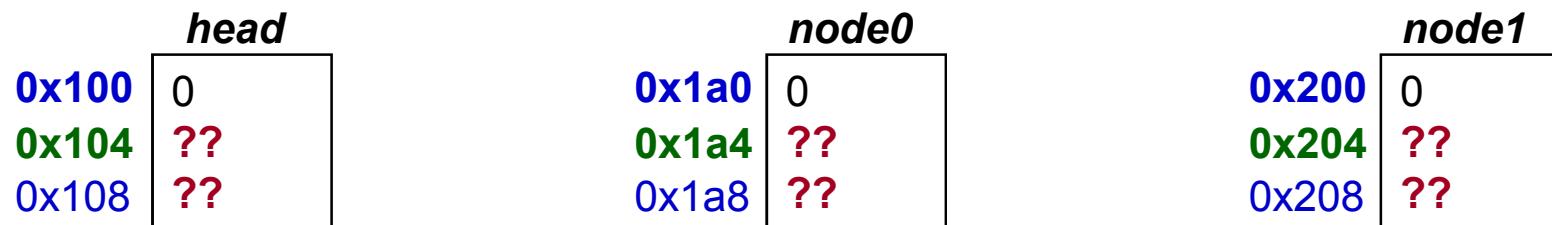


# Removing a Node



```
#define QREMOVE(node, alist)
 \
 \
 do {
 (1) *((node)->alist.prev) = (node)->alist.next;
 (2) ((node)->alist.next)->alist.prev = (node)->alist.prev;
 (3) (node)->alist.next = (node);
 (4) (node)->alist.prev = &((node)->alist.next);
 } while (/* */ 0)
```

QREMOVE(node0, alist);

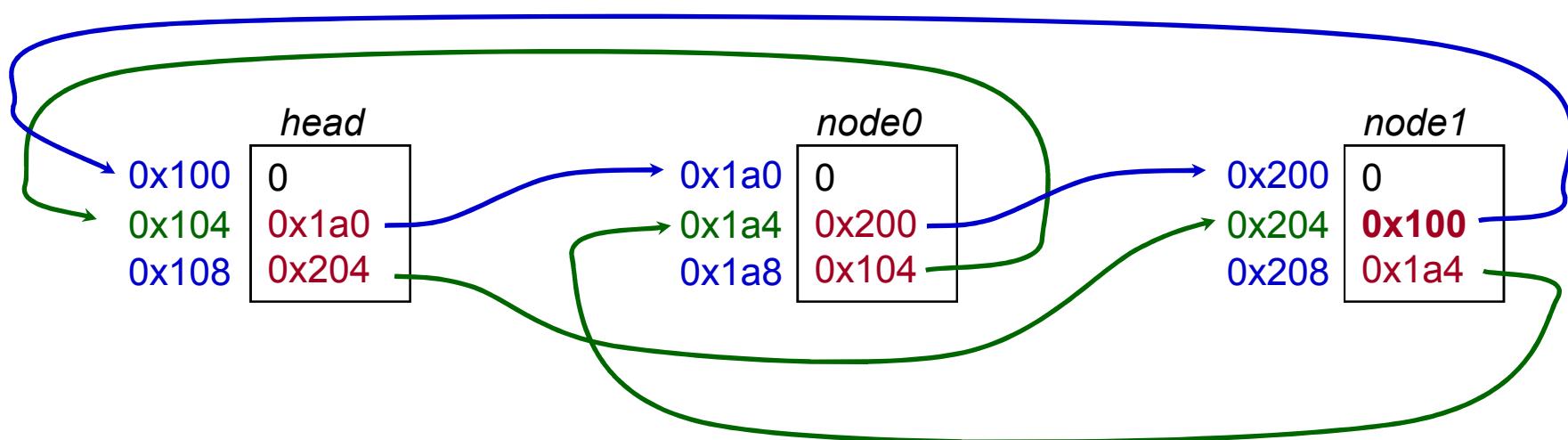


# Removing a Node

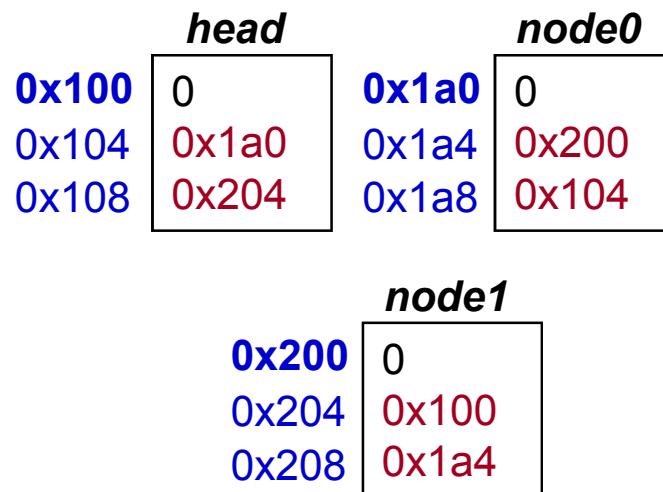
|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0            | 0x200        |
| 0x104 | 0x1a0       | 0x1a4        | 0x100        |
| 0x108 | 0x204       | 0x1a8        | 0x1a4        |

```
#define QREMOVE(node, alist)
 do {
 *((node)->alist.prev) = (node)->alist.next;
 ((node)->alist.next)->alist.prev = (node)->alist.prev;
 (node)->alist.next = (node);
 (node)->alist.prev = &((node)->alist.next);
 } while (/* */ 0)
```

QREMOVE(node0, alist);

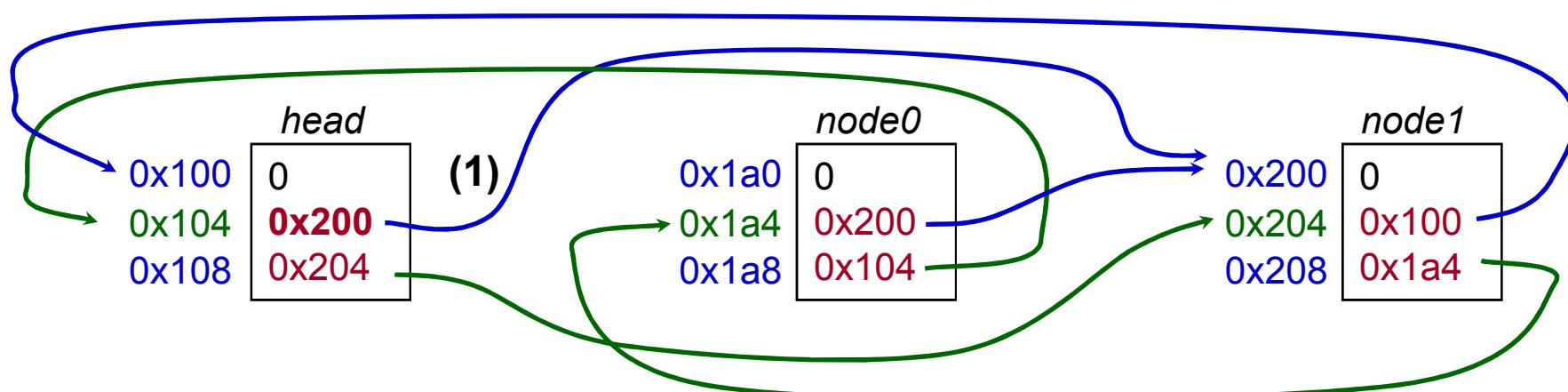


# Removing a Node



```
#define QREMOVE(node0, alist)
do {
 (1) *((node0)->alist.prev) = (node0)->alist.next; \
 ((node0)->alist.next)->alist.prev = (node0)->alist.prev; \
 (node0)->alist.next = (node0); \
 (node0)->alist.prev = &((node0)->alist.next); \
} while (/* */ 0)
```

QREMOVE(node0, alist);

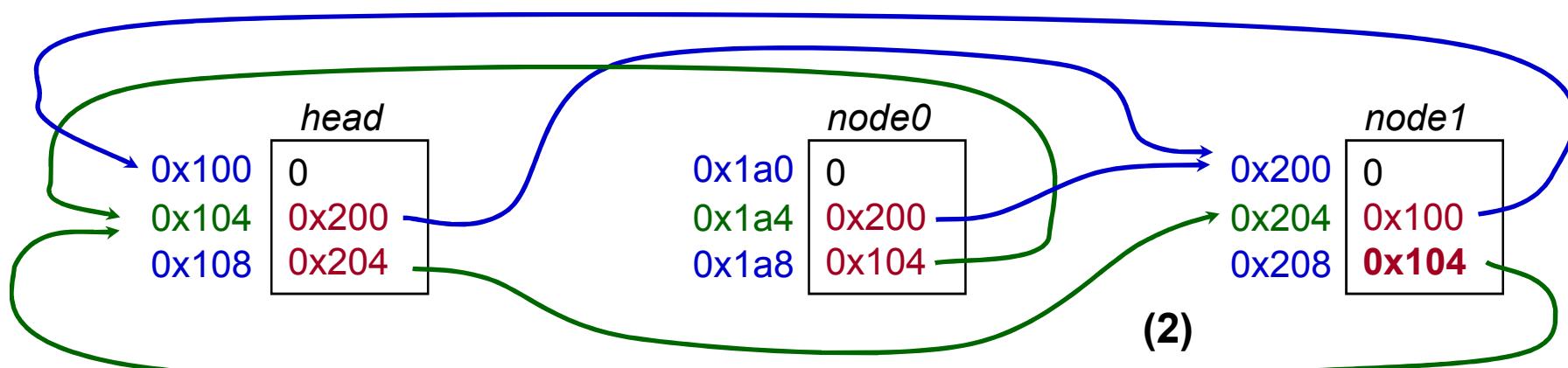


# Removing a Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0            | 0x200        |
| 0x104 | 0x1a0       | 0x200        | 0x100        |
| 0x108 | 0x204       | 0x104        | 0x1a4        |

```
#define QREMOVE(node0, alist)
 \
 \
 do {
 *((node0)->alist.prev) = (node0)->alist.next; \
 (2) ((node0)->alist.next)->alist.prev = (node0)->alist.prev; \
 (node0)->alist.next = (node0); \
 (node0)->alist.prev = &((node0)->alist.next); \
 } while (/* */ 0)
```

QREMOVE(node0, alist);

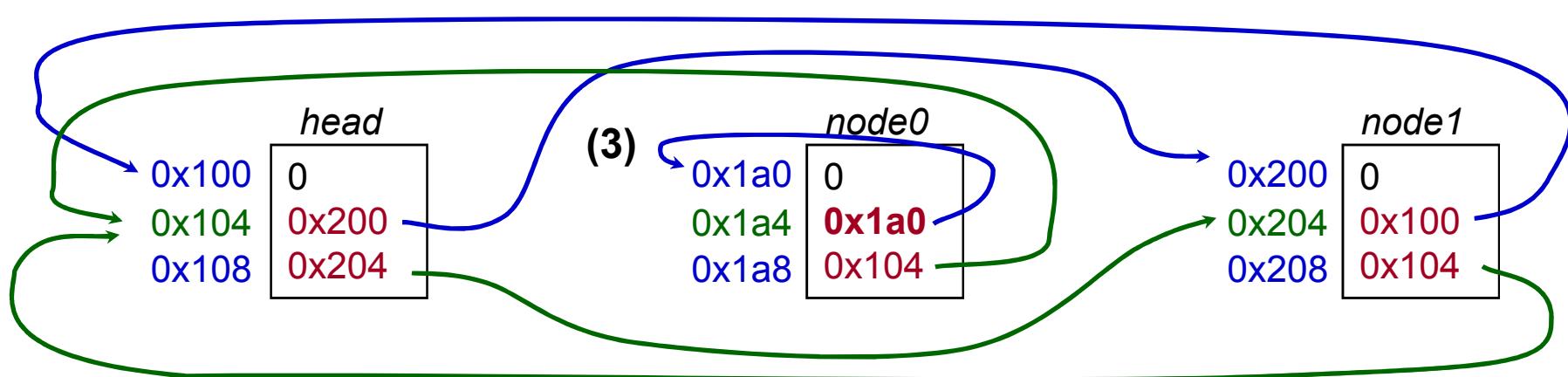


# Removing a Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0            | 0x200        |
| 0x104 | 0x1a0       | 0x200        | 0x100        |
| 0x108 | 0x204       | 0x104        | 0x1a4        |

```
#define QREMOVE(node0, alist) \
do { \
 *((node0)->alist.prev) = (node0)->alist.next; \
 ((node0)->alist.next)->alist.prev = (node0)->alist.prev; \
 (3) (node0)->alist.next = (node0); \
 (node0)->alist.prev = &((node0)->alist.next); \
} while (/* */ 0)
```

QREMOVE(node0, alist);

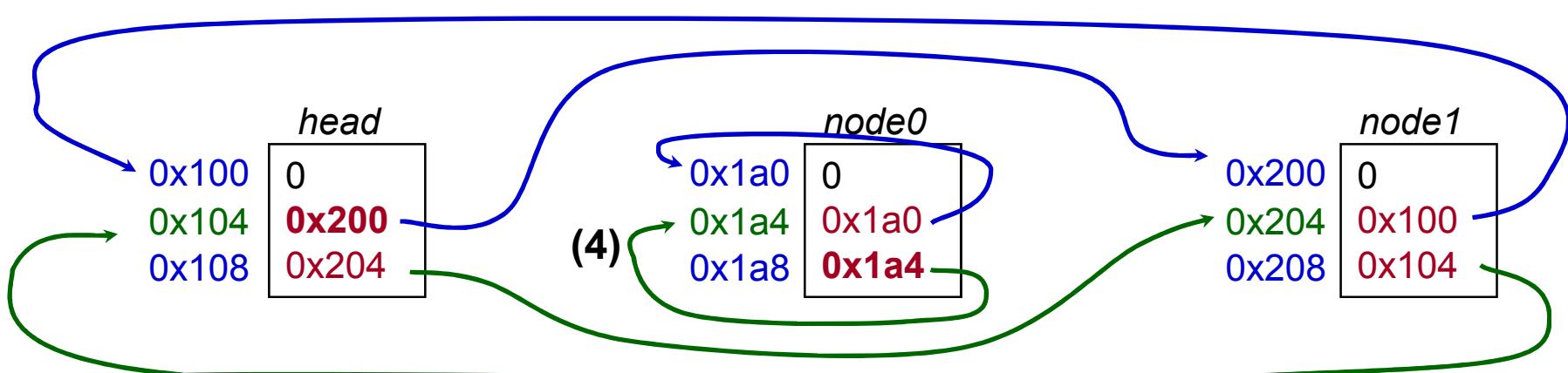


# Removing a Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0            | 0x200        |
| 0x104 | 0x1a0       | 0x200        | 0x100        |
| 0x108 | 0x204       | 0x104        | 0x1a4        |

```
#define QREMOVE(node0, alist)
 do {
 *((node0)->alist.prev) = (node0)->alist.next;
 ((node0)->alist.next)->alist.prev = (node0)->alist.prev;
 (node0)->alist.next = (node0);
 (4) (node0)->alist.prev = &((node0)->alist.next);
 } while (/* */ 0)
```

QREMOVE(node0, alist);

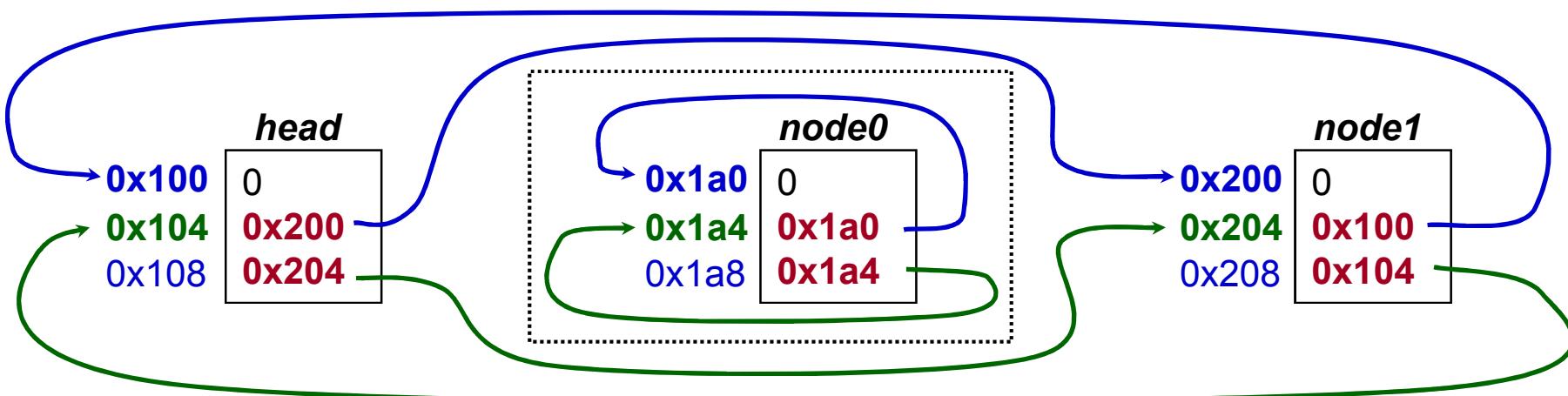


# Solution to Removing a Node

|       | <i>head</i> | <i>node0</i> | <i>node1</i> |
|-------|-------------|--------------|--------------|
| 0x100 | 0           | 0            | 0x200        |
| 0x104 | 0x1a0       | 0x200        | 0x100        |
| 0x108 | 0x204       | 0x104        | 0x1a4        |

```
#define QREMOVE(node, alist) \
 do { \
 (1) *((node)->alist.prev) = (node)->alist.next; \
 (2) ((node)->alist.next)->alist.prev = (node)->alist.prev; \
 (3) (node)->alist.next = (node); \
 (4) (node)->alist.prev = &((node)->alist.next); \
 } while (/* */ 0)
```

QREMOVE(node0, alist);



# Functions

- Always use function prototypes

```
int myfunc (char *, int, struct MyStruct *);
int myfunc_noargs (void);
void myfunc_noreturn (int i);
```
- C and C++ are *call by value*, copy of parameter passed to function
  - C++ permits you to specify pass by reference
  - if you want to alter the parameter then pass a pointer to it (or use references in C++)
- If performance is an issue then use inline functions, generally better and safer than using a macro. Common convention
  - define prototype and function in header or *name.i* file
  - static inline int myinfunc (int i, int j);
  - static inline int myinfunc (int i, int j) { ... }

# Basic Types and Operators

- Basic data types
  - Types: *char, int, float and double*
  - Qualifiers: *short, long, unsigned, signed, const*
- Constant: 0x1234, 12, "Some string"
- Enumeration:
  - Names in different enumerations must be distinct
  - `enum WeekDay_t {Mon, Tue, Wed, Thur, Fri};  
enum WeekendDay_t {Sat = 0, Sun = 4};`
- Arithmetic: +, -, \*, /, %
  - prefix ++i or --i ; increment/decrement before value is used
  - postfix i++, i--; increment/decrement after value is used
- Relational and logical: <, >, <=, >=, ==, !=, &&, ||
- Bitwise: &, |, ^ (xor), <<, >>, ~(ones complement)

# Operator Precedence

(from "C a Reference Manual", 5<sup>th</sup> Edition)

| Tokens                 | Operator                            | Class   | Precedence | Associates    |
|------------------------|-------------------------------------|---------|------------|---------------|
| <i>names, literals</i> | simple tokens                       | primary |            | n/a           |
| <b>a[k]</b>            | subscripting                        | postfix |            | left-to-right |
| <b>f(...)</b>          | function call                       | postfix |            | left-to-right |
| .                      | direct selection                    | postfix | 16         | left-to-right |
| ->                     | indirect selection                  | postfix |            | left-to-right |
| <b>++ --</b>           | increment, decrement                | postfix |            | left-to-right |
| <b>(type) {init}</b>   | compound literal                    | postfix |            | left-to-right |
| <b>++ --</b>           | increment, decrement                | prefix  |            | right-to-left |
| <b>sizeof</b>          | size                                | unary   |            | right-to-left |
| <b>~</b>               | bitwise not                         | unary   |            | right-to-left |
| <b>!</b>               | logical not                         | unary   | 15         | right-to-left |
| <b>- +</b>             | negation, plus                      | unary   |            | right-to-left |
| <b>&amp;</b>           | address of                          | unary   |            | right-to-left |
| <b>*</b>               | indirection<br><i>(dereference)</i> | unary   |            | right-to-left |

| Tokens                       | Operator          | Class   | Precedence | Associates    |
|------------------------------|-------------------|---------|------------|---------------|
| <b>(type)</b>                | casts             | unary   | 14         | right-to-left |
| <b>* / %</b>                 | multiplicative    | binary  | 13         | left-to-right |
| <b>+ -</b>                   | additive          | binary  | 12         | left-to-right |
| <b>&lt;&lt; &gt;&gt;</b>     | left, right shift | binary  | 11         | left-to-right |
| <b>&lt; &lt;= &gt; &gt;=</b> | relational        | binary  | 10         | left-to-right |
| <b>== !=</b>                 | equality/ineq.    | binary  | 9          | left-to-right |
| <b>&amp;</b>                 | bitwise and       | binary  | 8          | left-to-right |
| <b>^</b>                     | bitwise xor       | binary  | 7          | left-to-right |
| <b> </b>                     | bitwise or        | binary  | 6          | left-to-right |
| <b>&amp;&amp;</b>            | logical and       | binary  | 5          | left-to-right |
| <b>  </b>                    | logical or        | binary  | 4          | left-to-right |
| <b>? :</b>                   | conditional       | ternary | 3          | right-to-left |
| <b>= += -= *= /= %=</b>      | assignment        | binary  | 2          | right-to-left |
| <b>&lt;&lt;= &gt;&gt;=</b>   |                   |         |            |               |
| ,                            | sequential eval.  | binary  | 1          | left-to-right |

# Structs and Unions

- **Structures**
  - `struct MyPoint {int x, int y};`
  - `typedef struct MyPoint MyPoint_t;`
  - `MyPoint_t point, *ptr;`
  - `point.x = 0; point.y = 10;`
  - `ptr = &point; ptr->x = 12; ptr->y = 40;`
- **Unions**
  - `union MyUnion {int x; MyPoint_t pt; struct {int  
    3; char c[4]} S;};`
  - `union MyUnion x;`
  - **Can only use one of the elements. Memory will be allocated for the largest element**

# Conditional Statements (if/else)

```
if (a < 10)
 printf("a is less than 10\n");
else if (a == 10)
 printf("a is 10\n");
else
 printf("a is greater than 10\n");
```

- If you have compound statements then use brackets (blocks)
  - ```
if (a < 4 && b > 10) {
    c = a * b; b = 0;
    printf("a = %d, a's address = 0x%08x\n", a, (uint32_t)&a);
} else {
    c = a + b; b = a;
}
```
- These two statements are equivalent:
 - ```
if (a) x = 3; else if (b) x = 2; else x = 0;
```
  - ```
if (a) x = 3; else {if (b) x = 2; else x = 0; }
```
- Is this correct?
 - ```
if (a) x = 3; else if (b) x = 2;
else (z) x = 0; else x = -2;
```

# Conditional Statements (switch)

```
int c = 10;
switch (c) {
 case 0:
 printf("c is 0\n");
 break;
 ...
 default:
 printf("Unknown value of c\n");
 break;
}
```

- What if we leave the break statement out?
- Do we need the final break statement on the default case?

# Loops

```
for (i = 0; i < MAXVALUE; i++) {
 dowork();
}

while (c != 12) {
 dowork();
}

do {
 dowork();
} while (c < 12);
```

- flow control
  - **break** - exit innermost loop
  - **continue** - perform next iteration of loop
- Note, all these forms permit one statement to be executed. By enclosing in brackets we create a block of statements.

# Building your program

- For all labs and programming assignments:
  - you must supply a make file
  - you must supply a README file that describes the assignment and results. This must be a text file, no MS word.
  - of course the source code and any other libraries or utility code you used
  - you may submit plots, they must be postscript or pdf

# make and Makefile - Overview

- Why use make?
  - convenience of only entering compile directives once
  - make is smart enough (with your help) to only compile and link modules that have changed or which depend on files that have changed
  - allows you to hide platform dependencies
  - promotes uniformity
  - simplifies my (and hopefully your) life when testing and verifying your code
- A Makefile contains a set of rules for building a program
  - target ... : prerequisites ...
    - command
    - ...
- Static pattern rules.
  - each target is matched against target-pattern to derive stem which is used to determine prereqs (see example)
    - targets ... : target-pattern : prereq-patterns ...
      - command
      - ...

# Makefiles

- **Defining variables**

```
MyOPS := -DWTH
```

```
MyDIR ?= /home/fred
```

```
MyVar = $(SHELL)
```

- **Using variables**

```
MyFLAGS := $(MyOPS)
```

- **Built-in Variables**

- \$@ = filename of target

- \$< = name of the first prerequisites

- **Patterns**

- use % character to determine stem

- foo.o matches the pattern %.o with foo as the stem.

- foo.o moo.o : %.o : %.c # says that foo.o depends on foo.c and moo.o depends on moo.c

# Example Makefile for wulib

## Makefile.inc

```
Makefile.inc
Contains common definitions

MyOS := $(shell uname -s)
MyID := $(shell whoami)
MyHost := $(shell hostname)
WARNSTRICT := -W \
 -Wstrict-prototypes
\ -Wmissing-prototypes
WARNLIGHT := -Wall
WARN := ${WARNLIGHT}
ALLFLGS := -D_GNU_SOURCE \
 -D_REENTRANT \
 -D_THREAD_SAFE
APPCFLAGS = ${ALLFLGS} \
 ${WARN}

WUCC := gcc
WUCFLAGS := -DMyOS=${MyOS} \
 ${OSFLAGS} \
 ${ALLFLGS} ${WARN}

WUINCLUDES :=
WULIBS := -lm

ifeq (${MyOS}, SunOS)
OSLIBS+= -lrt
endif
```

## Makefile

```
Project specific
include ./Makefile.inc
INCLUDES = ${WUINCLUDES} -I.
LIBS = ${WILIBS} ${OSLIBS}
CFLAGS = ${WUCFLAGS} -DWUDEBUG
CC = ${WUCC}

HDRS := util.h
CSRCS := testapp1.c testapp2.c
SRCS := util.c callout.c
COBJS = $(addprefix ${OBJDIR}/, \
 $(patsubst %.c,%.o,$(CSRCS)))
OBJS = $(addprefix ${OBJDIR}/, \
 $(patsubst %.c,%.o,$(SRCS)))
CMDS = $(addprefix ${OBJDIR}/, $(basename $(CSRCS)))

all : ${OBJDIR} ${CMDS}

install : all

${OBJDIR} :
 mkdir ${OBJDIR}

${OBJS} ${COBJS} : ${OBJDIR}/%.o : %.c ${HDRS}
 ${CC} ${CFLAGS} ${INCLUDES} -o $@ -c $<

${CMDS} : ${OBJDIR}/% : ${OBJDIR}/%.o ${OBJS}
 ${CC} ${CFLAGS} -o $@ $@.o ${LIBS}
 chmod 0755 $@

clean :
 /bin/rm -f ${CMDS} ${OBJS}
```

# Project Documentation

README file structure

## ***Section A: Introduction***

describe the project, paraphrase the requirements and state your understanding of the assignments value.

## ***Section B: Design and Implementation***

List all files turned in with a brief description for each. Explain your design and provide simple psuedo-code for your project. Provide a simple flow chart of you code and note any constraints, invariants, assumptions or sources for reused code or ideas.

## ***Section C: Results***

For each project you will be given a list of questions to answer, this is where you do it. If you are not satisfied with your results explain why here.

## ***Section D: Conclusions***

What did you learn, or not learn during this assignment. What would you do differently or what did you do well.

# Attacking a Project

**Requirements and scope:** Identify specific requirements and or goals. Also note any design and/or implementation environment requirements.

knowing when you are done, or not done

estimating effort or areas which require more research

programming language, platform and other development environment issues

**Approach:** How do you plan to solve the problem identified in the first step. Develop a prototype design and document. Next figure out how you will verify that you did satisfy the requirements/goals. Designing the tests will help you to better understand the problem domain and your proposed solution

**Iterative development:** It is good practice to build your project in small pieces. Testing and learning as you go.

**Final Touches:** Put it all together and run the tests identified in the approach phase. Verify you met requirements. Polish you code and documentation.

Turn it in.

# Java Programming

# Running Java Code

| command | arguments                                                     | purpose              |
|---------|---------------------------------------------------------------|----------------------|
| javac   | .java file name                                               | compile Java program |
| java    | .class file name (no extension)<br>and command-line arguments | run Java program     |

# Java programming

Install java (use: **open jdk, avoid Oracle java**)  
in easy-to-type toplevel directory (name  
without spaces), e.g.

c:\java\jdk16

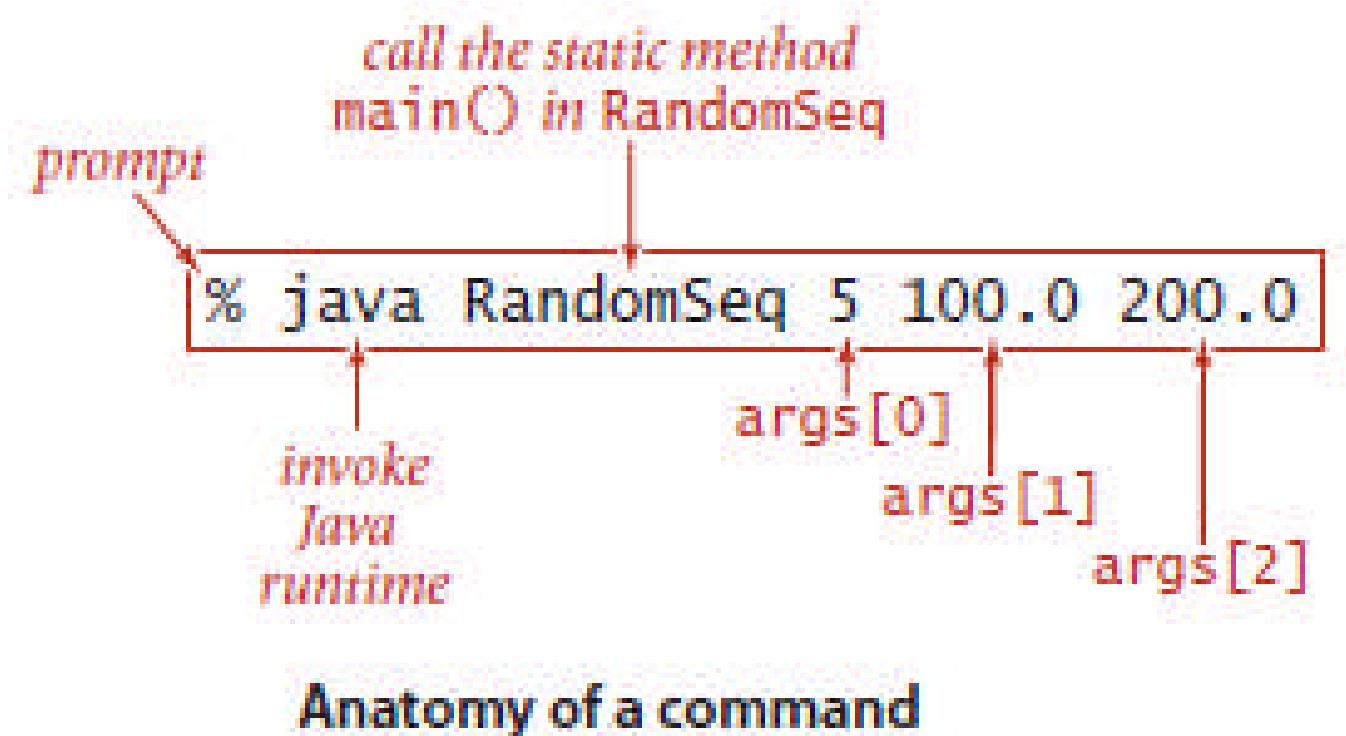
c:\java\jre16

**To compile and run a java file:**

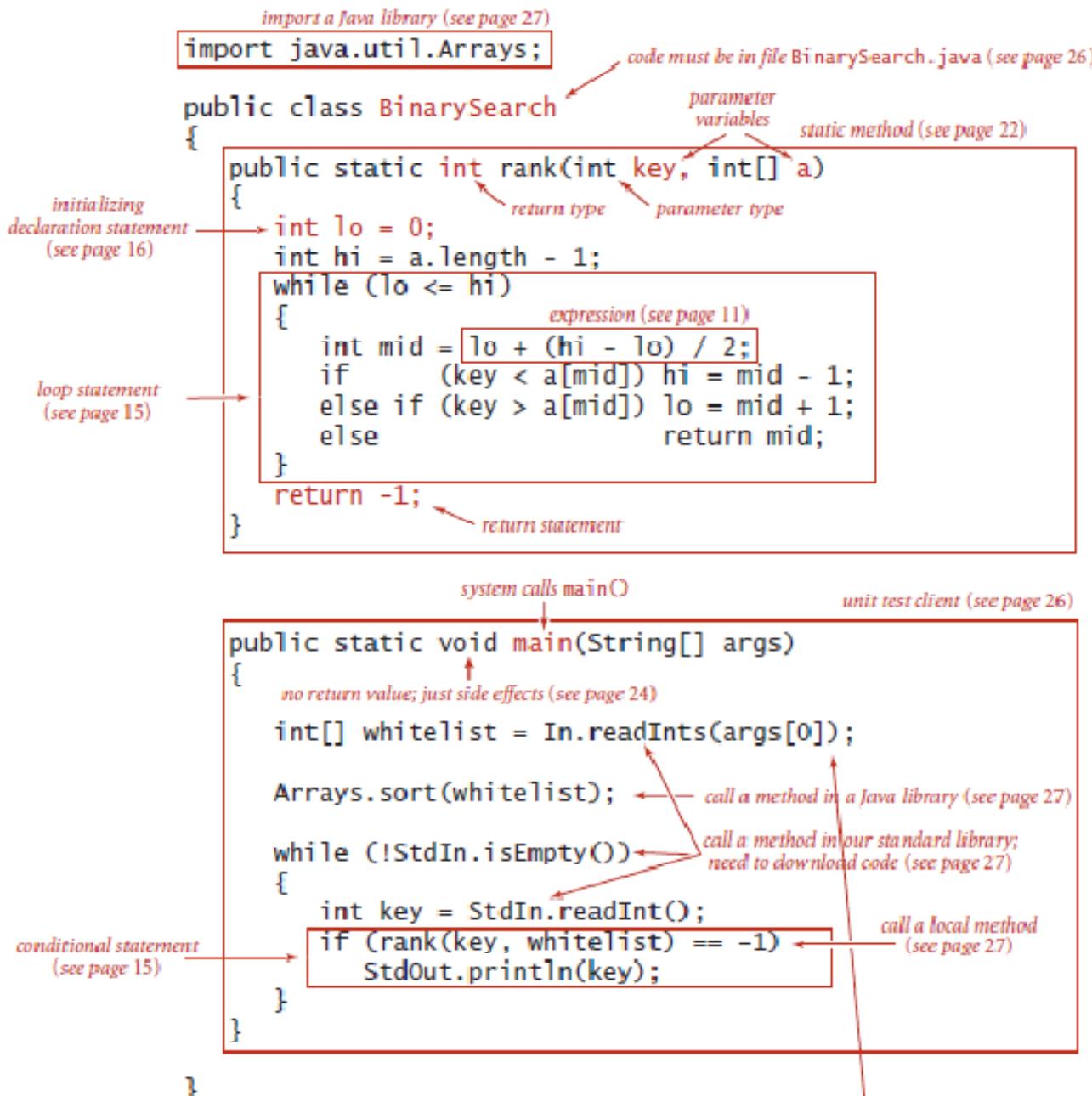
C:\> javac BinarySearch.java

C:\> java -cp . BinarySearch < input.txt

# Running a Java class

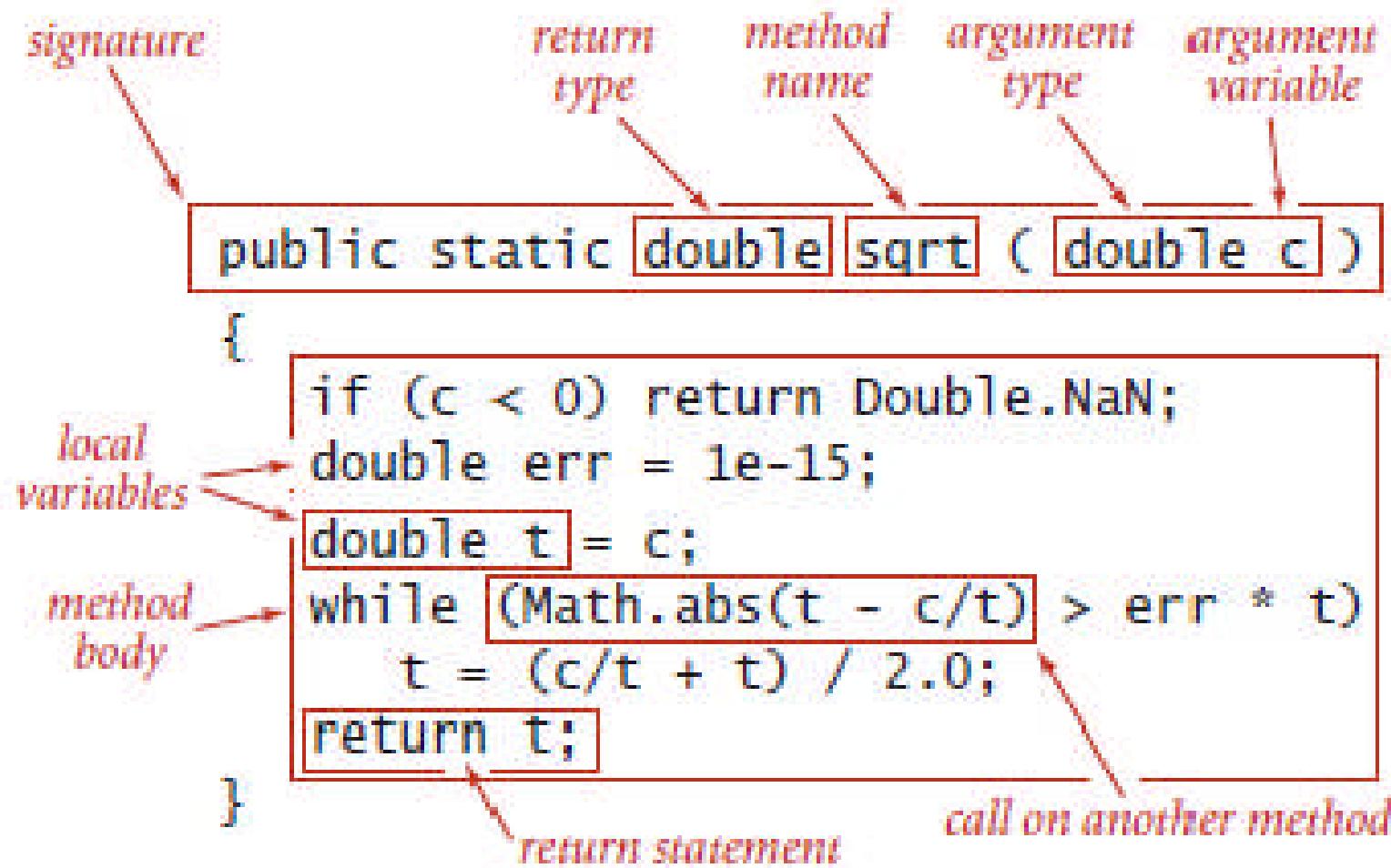


# Java Class Example:



BinarySearch.java

# Java functions syntax



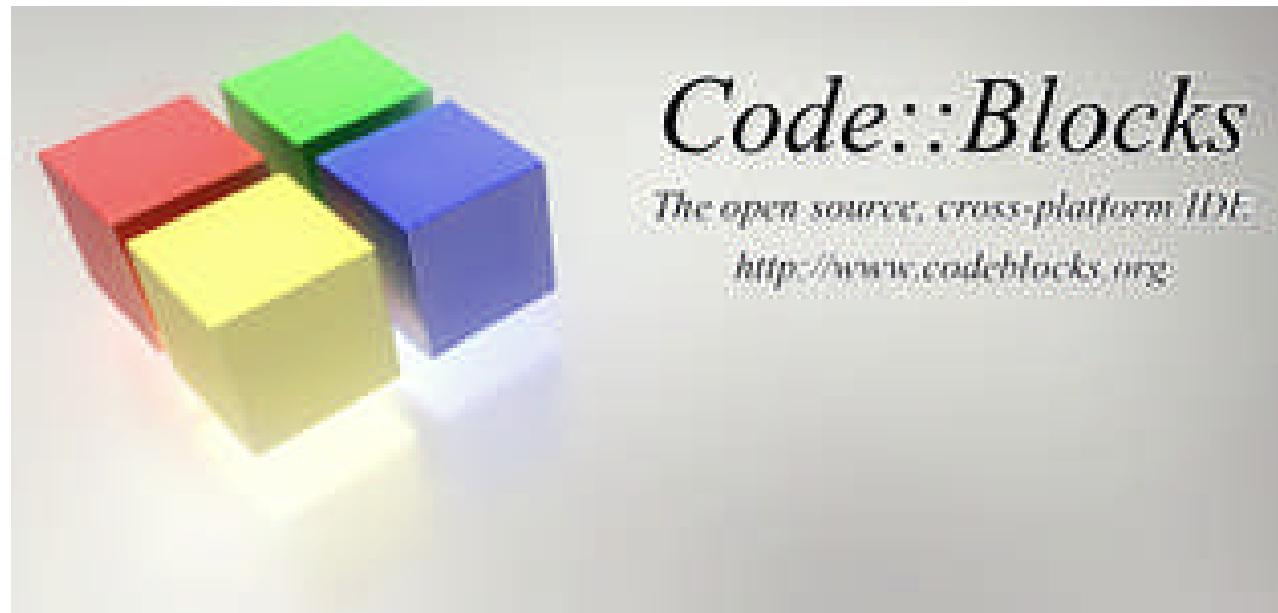
Anatomy of a static method

# References

1. Book "Core Java", Vol 1 and 2,  
by Horstmann and Cornell  
website: <http://www.horstmann.com/corejava.html>

# Using Code::Blocks IDE

(Integrated Development Environment)



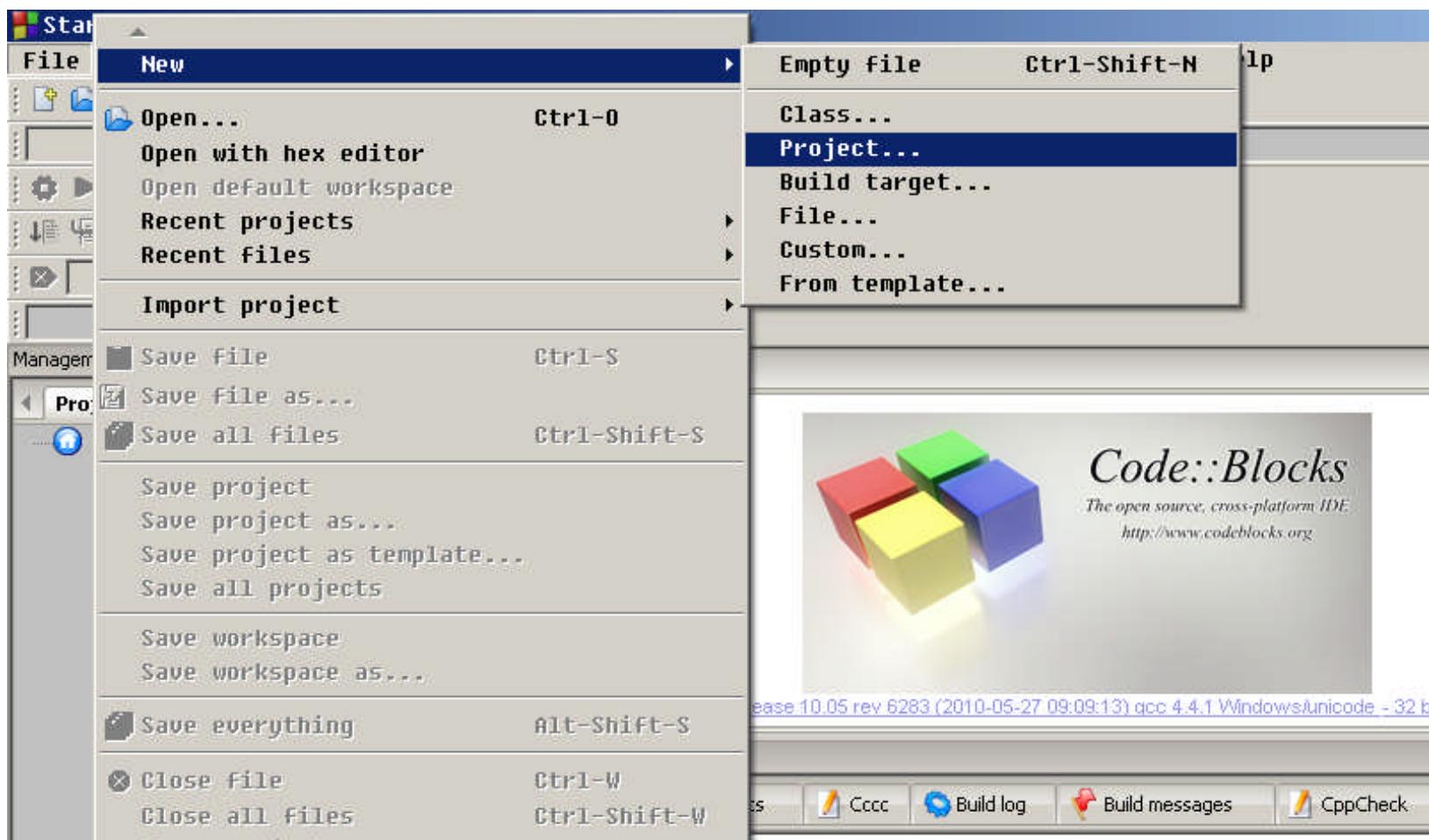
# Install Codeblocks

- Codeblocks is the free IDE for C/C++ programming, it comes with the gcc and g++ compilers and libraries.
- It works well on Windows. Windows gcc is called mingw gcc
- Download and Install Codeblocks (with gcc compiler) on your computer, e.g. in `c:/tools/codeblocks` (no spaces in path).

# Run Codeblock

- Run Codeblocks, then click on the menu:
- File > New > Project > Console Application > Select C or C++ application >
- Pick a new folder to create your application, give the directory a good name (without spaces in path), so you can find it again.
- Click next to finish the wizard

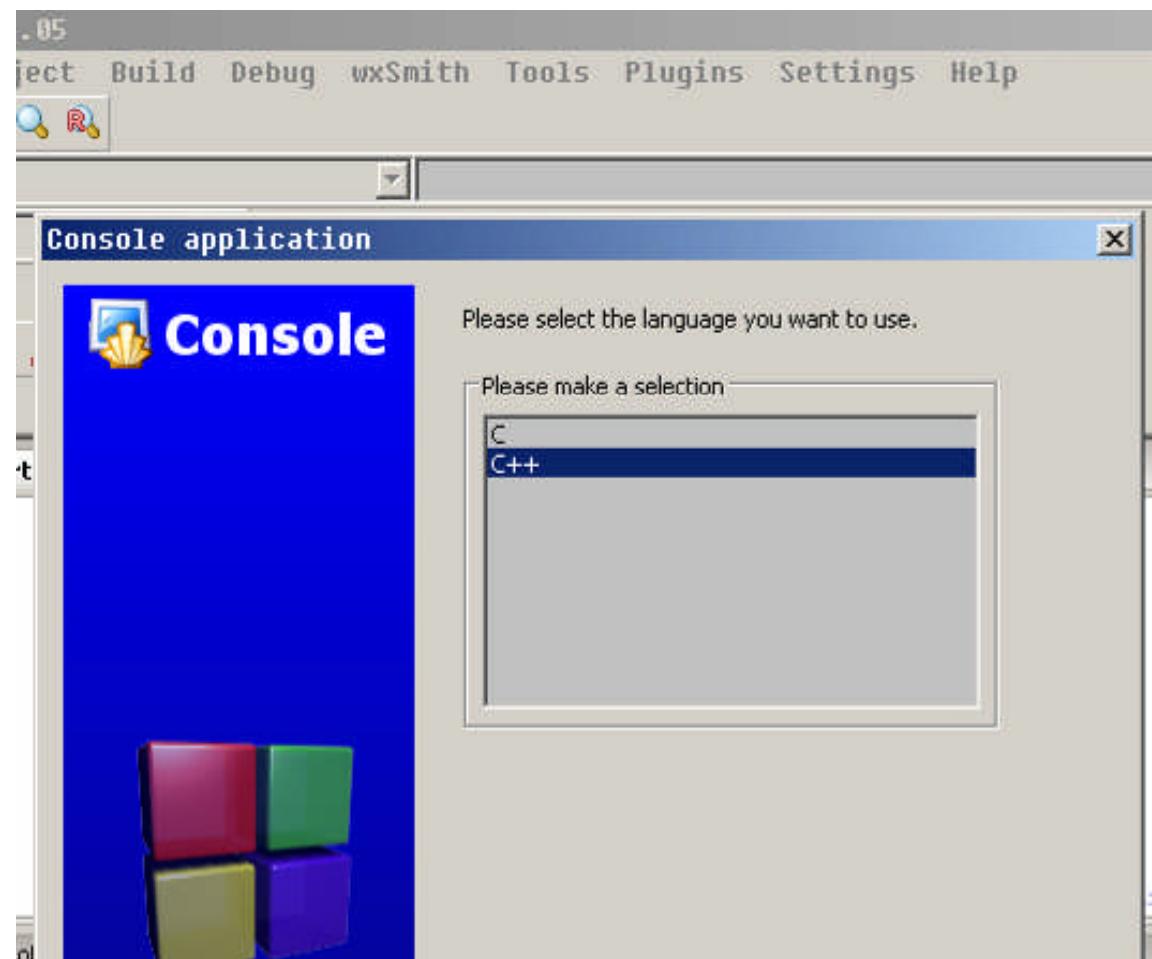
# Create project



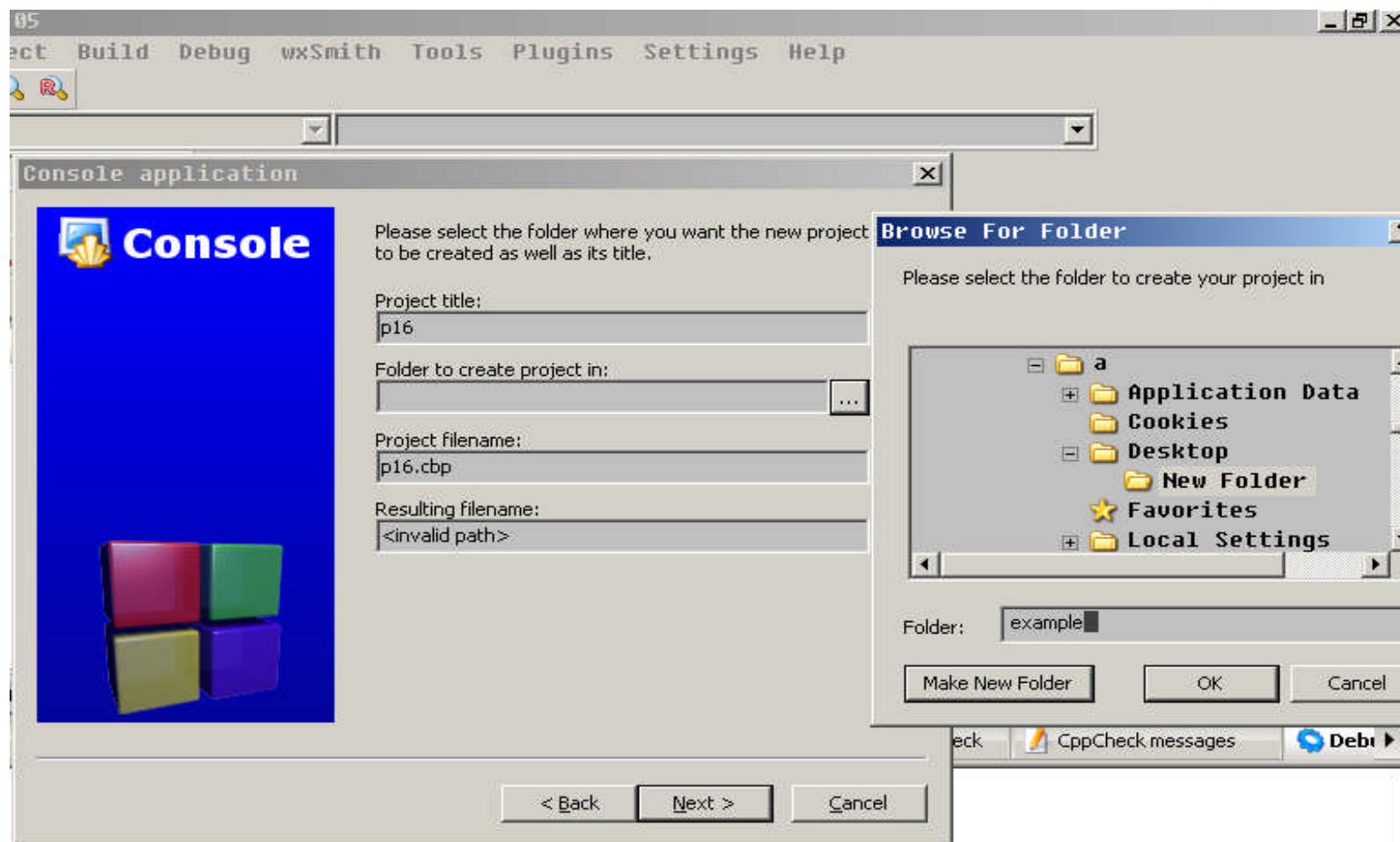
# Console application



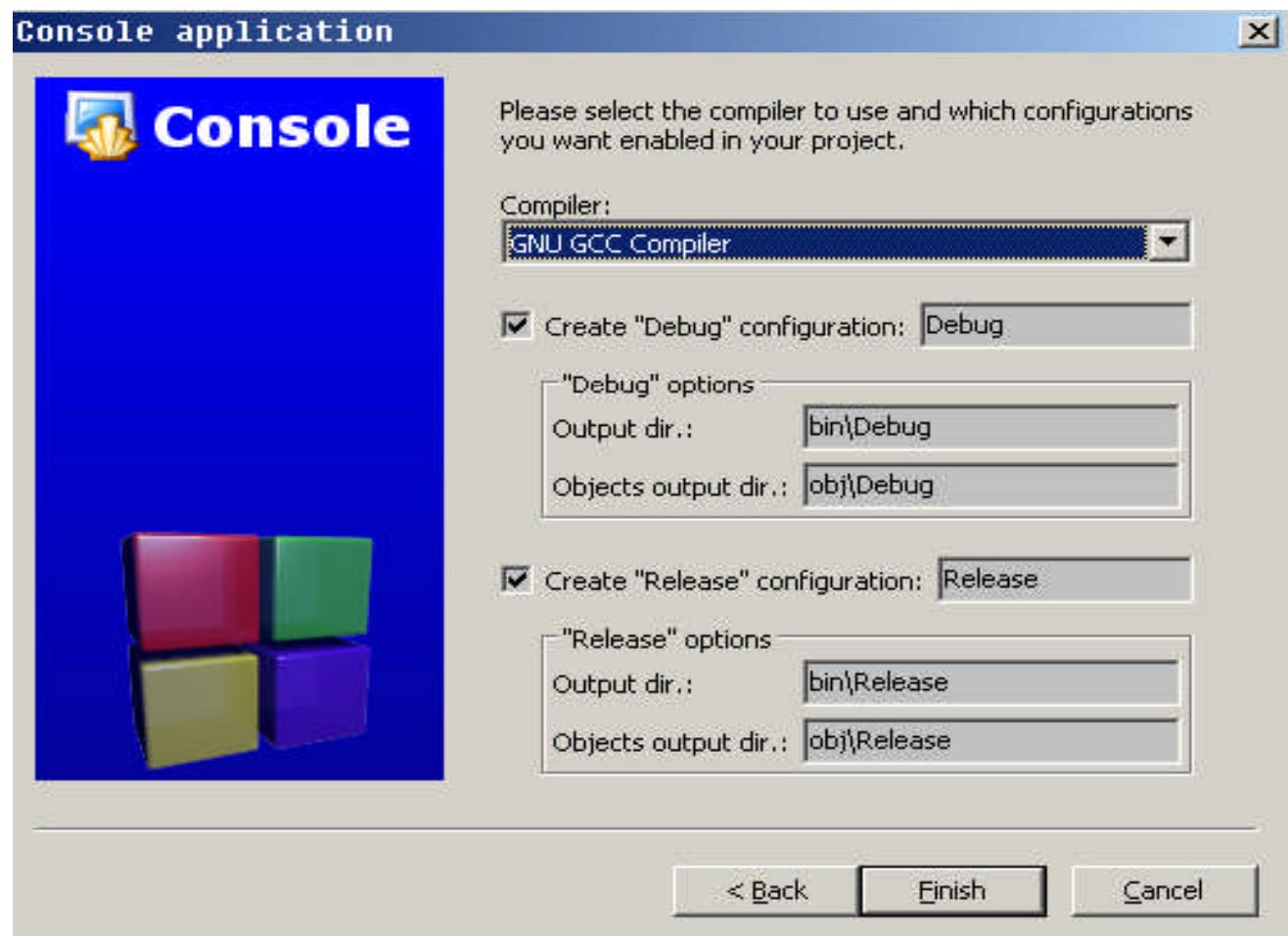
# Type of program



# Directory to save files



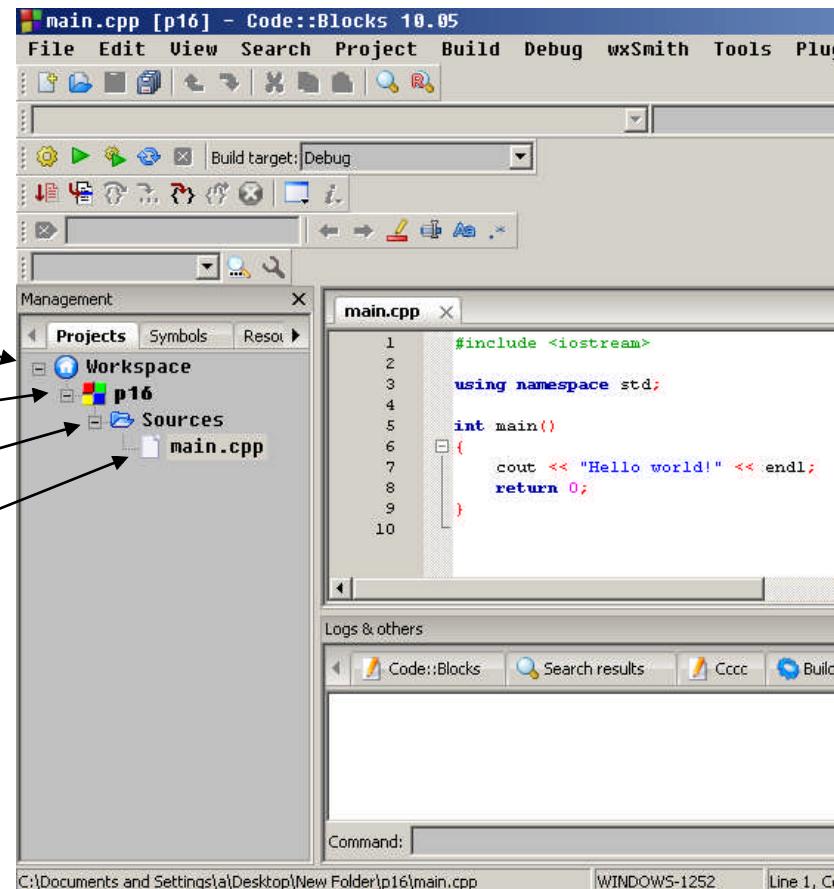
# Compiler to use



# Workspace

Now you can start  
programming,  
click on the left  
pane on

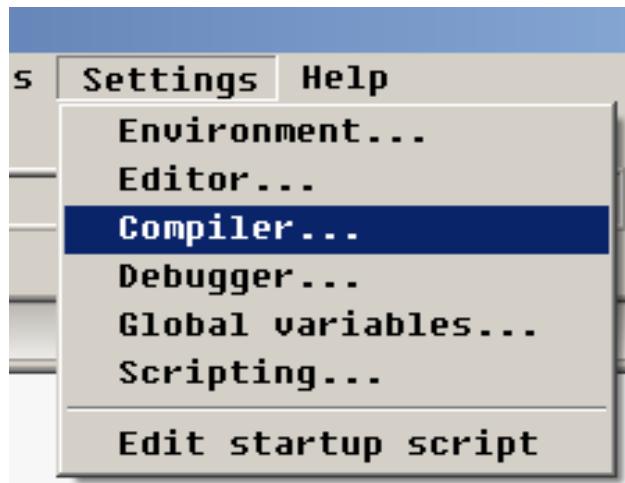
- Workspace >
- project name >
- Sources >
- main.cpp



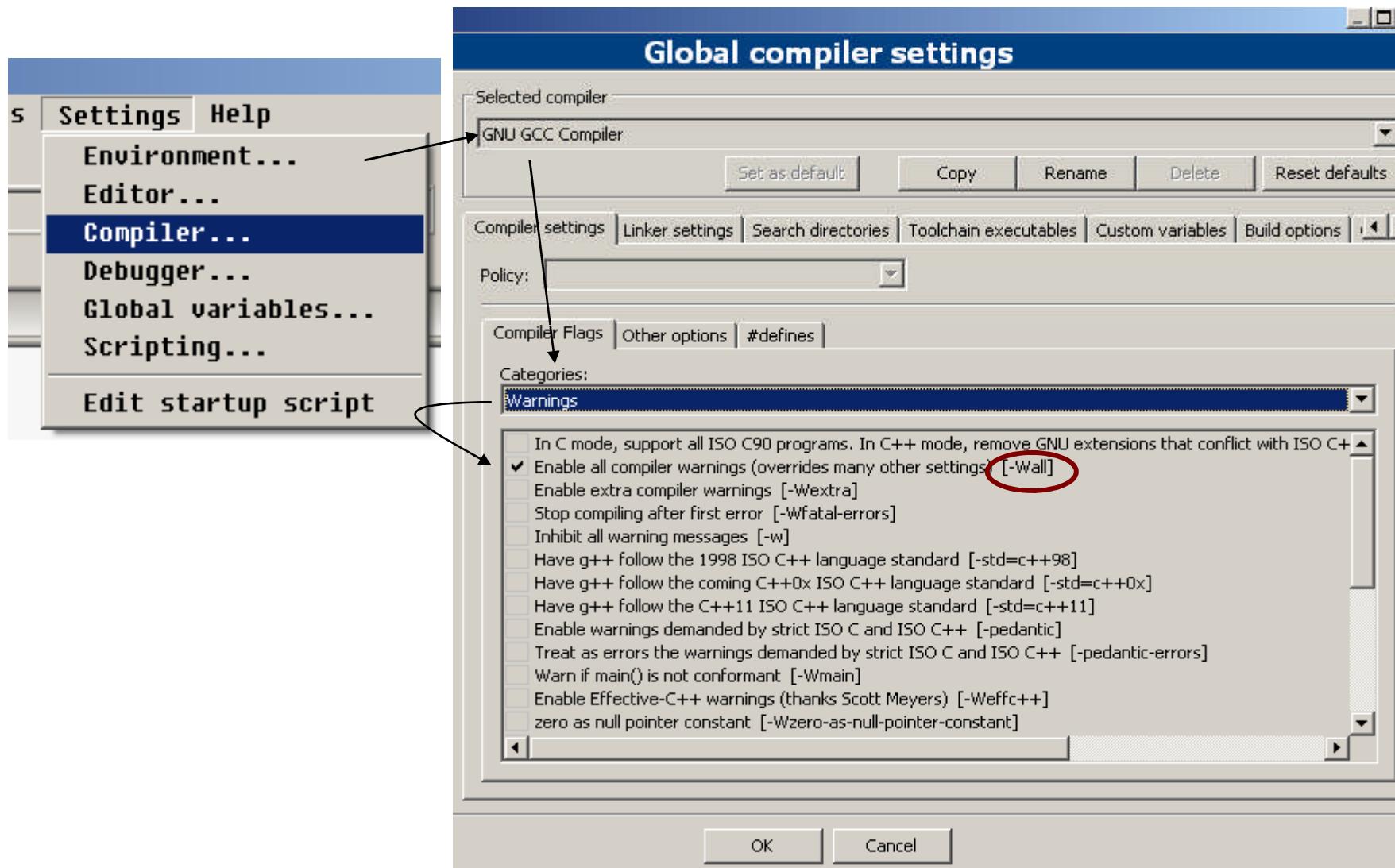
# Type the *hello world* program, main.cpp

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4. cout << "Hello world!" << endl;
5. return 0;
6. }
```

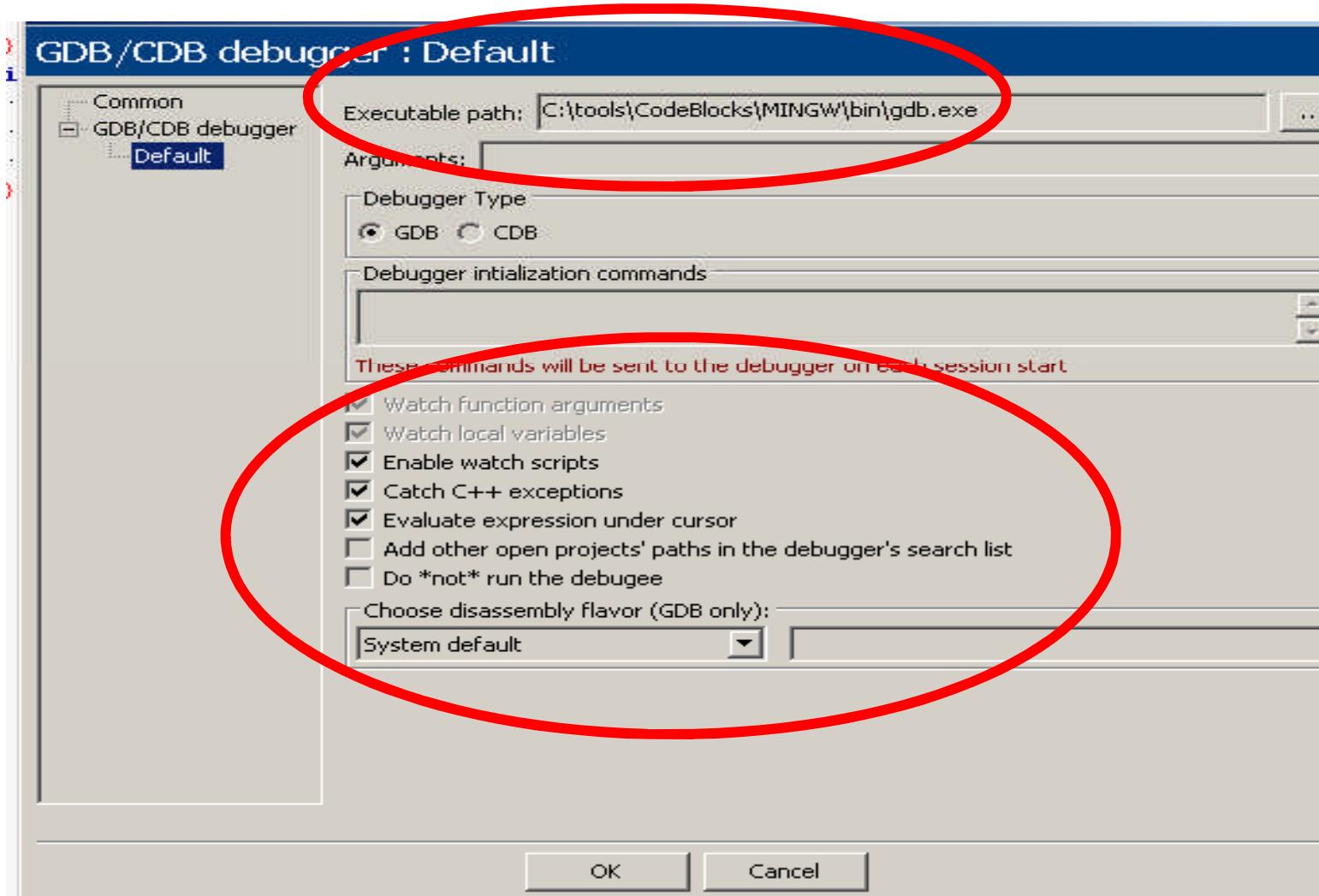
# Settings>Compiler: enable -g for debug symbols



# Settings>Compiler: enable –Wall to see all warnings during compiling.

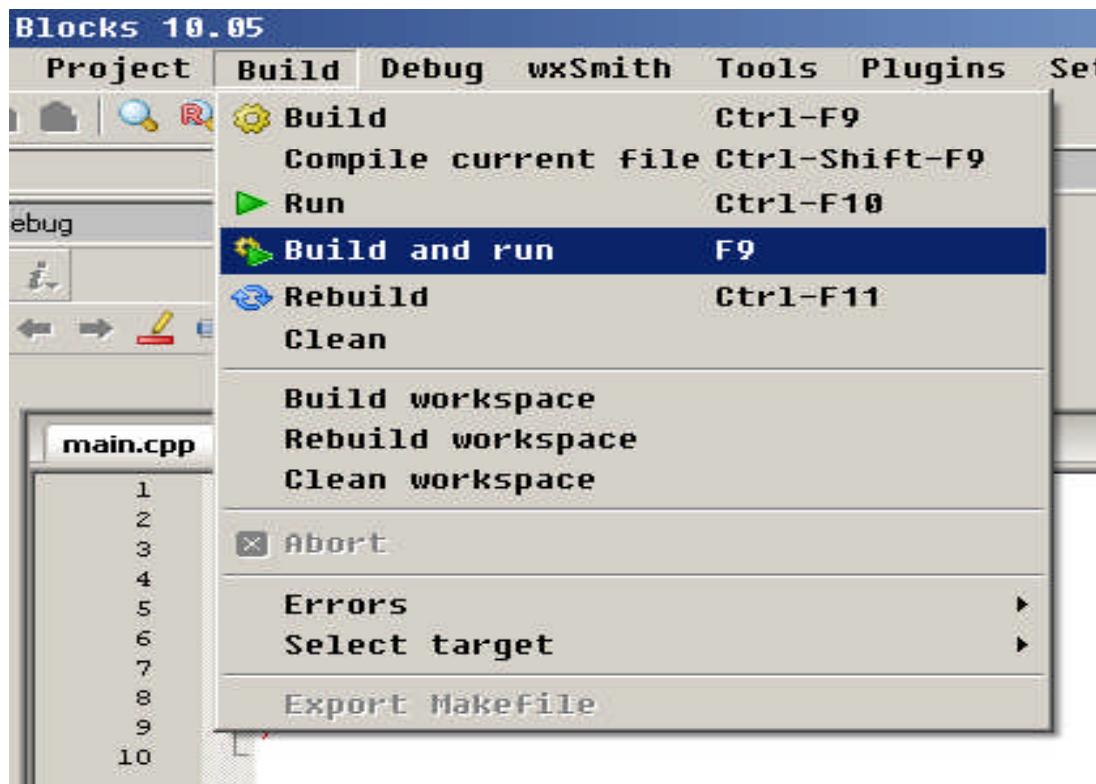


# Settings > Debugger



# Build and run

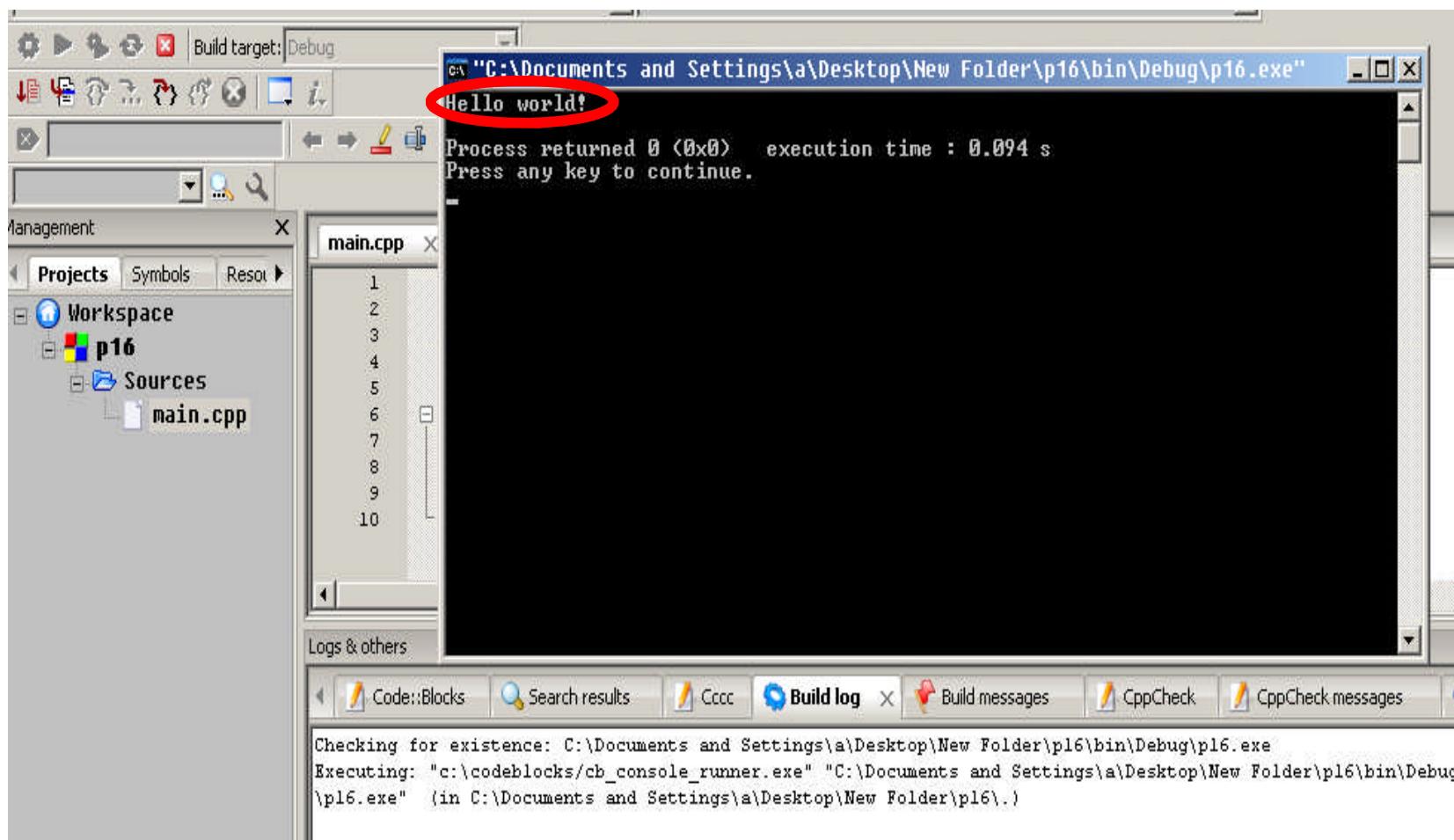
- Now we can run the hello-world console application:



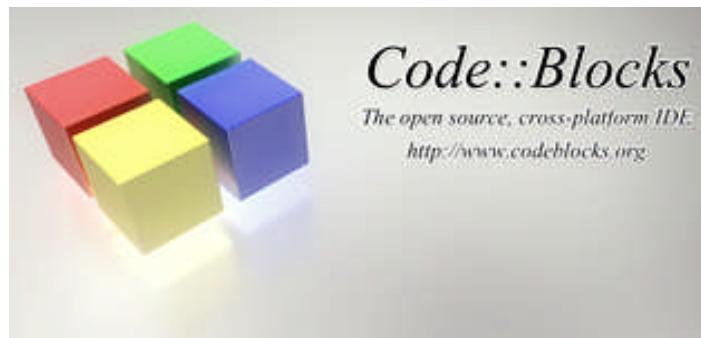
# Compiler output

- You will see the compiler output in the build log pane on the bottom right window.
- If there are no errors, the program will run and a *console* window will pop up, with the output “Hello world!”.

# Run the program



# Debugging with Codeblocks



# Why debug?

- Programming without a debugger is like driving a car without knowing how to use brakes.
- Whenever there is a problem, you just break (brake) and see what is wrong in the source code, along with the variable values. It is much easier than guessing what is wrong by staring at the code.

# Before you debug

1. Compile your code with gcc –Wall and fix all warnings.
2. Add assertions to the code to check your assumptions. E.g.

```
#include <assert.h>

...
g = gcd(3,7);
assert(g == 1);

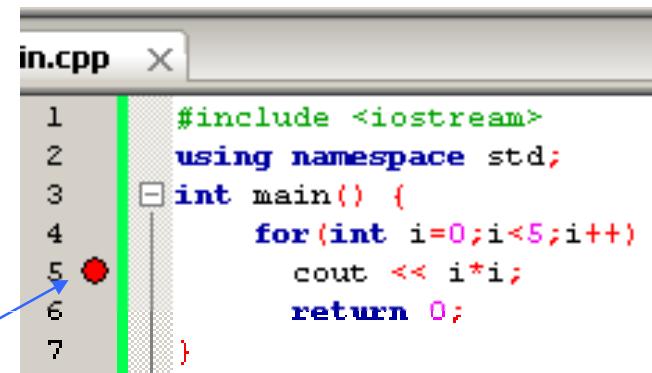
...
```

3. Use the latest Codeblocks (with gcc).
4. Don't use obsolete Turbo-C or Dev-Cpp.

# Example C++ program

- Type the following program into codeblock main.cpp

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4. for(int i=0;i<5;i++)
5. cout << i*i;
6. return 0;
7. }
```



A screenshot of the Code::Blocks IDE showing a file named "in.cpp". The code is identical to the one above. A blue arrow points from the text "Click next to the number '5' to create a breakpoint (red dot in the image)." to the line number 5 in the editor. A red dot is visible on the left margin next to the line number 5, indicating a breakpoint.

```
#include <iostream>
using namespace std;
int main() {
 for(int i=0;i<5;i++)
 cout << i*i;
 return 0;
}
```

- Click next to the number '5' to create a breakpoint (red dot in the image).
- Then from the Build menu, select build to compile the program and ensure there are no errors.
- If there is an error, you will see a red mark and the error message will be shown in the build-log.

# Build main.cpp

The screenshot shows a C++ development environment with the following interface elements:

- Projects View:** Shows a workspace named "p16" containing a source file "main.cpp".
- Code Editor:** A tab labeled "main.cpp" displays the following code:

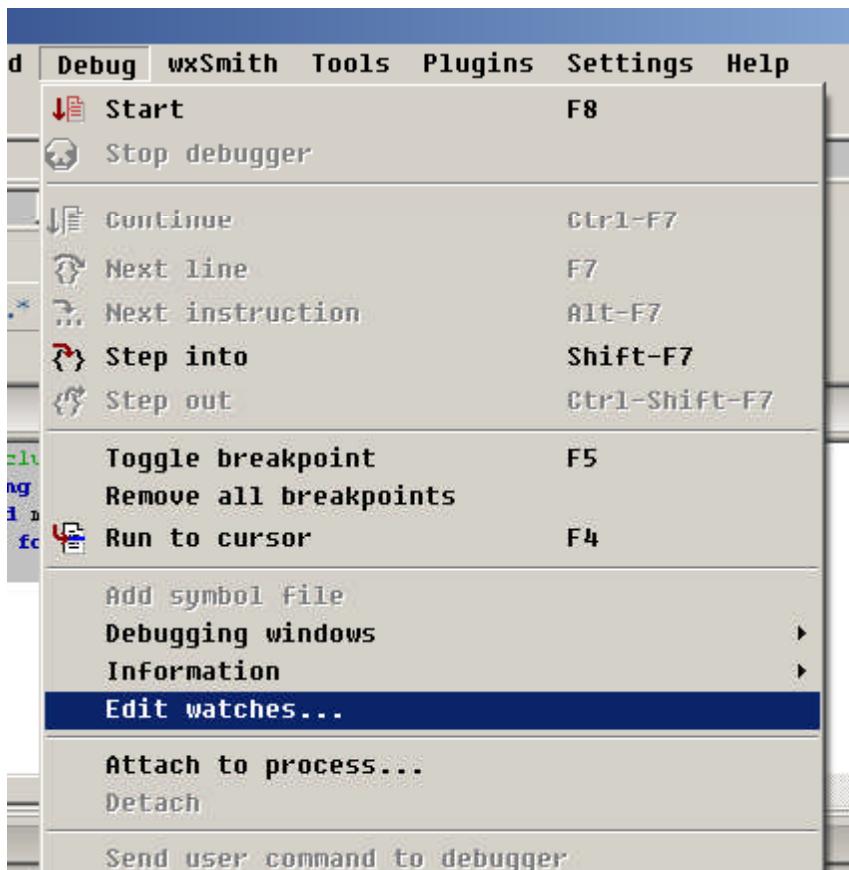
```
1 #include <iostream>
2 using namespace std;
3 int main() {
4 for(int i=0;i<5;i++)
5 cout << i*i;
6 return 0;
7 }
```

A red dot marks the cursor at the end of line 5.
- Logs & others:** A panel at the bottom right contains tabs for "Search results", "Cccc", "Build log", and "Build messages".
- Build Log Output:** The "Build log" tab shows the following build logs:

```
----- Build: Debug in p16 -----
Compiling: main.cpp
Linking console executable: bin\Debug\p16.exe
Output size is 913.06 KB
Process terminated with status 0 (0 minutes, 2 seconds)
0 errors, 0 warnings
```

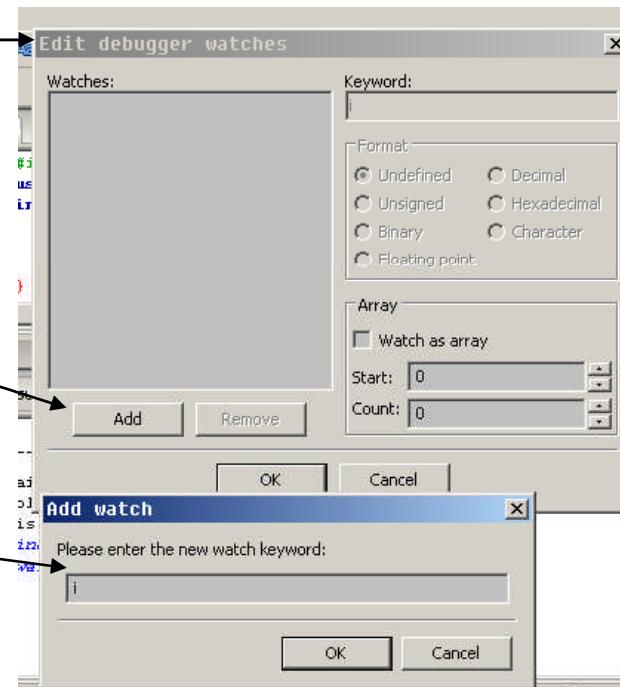
# Watch variables

- Click on ‘debug > edit watches’

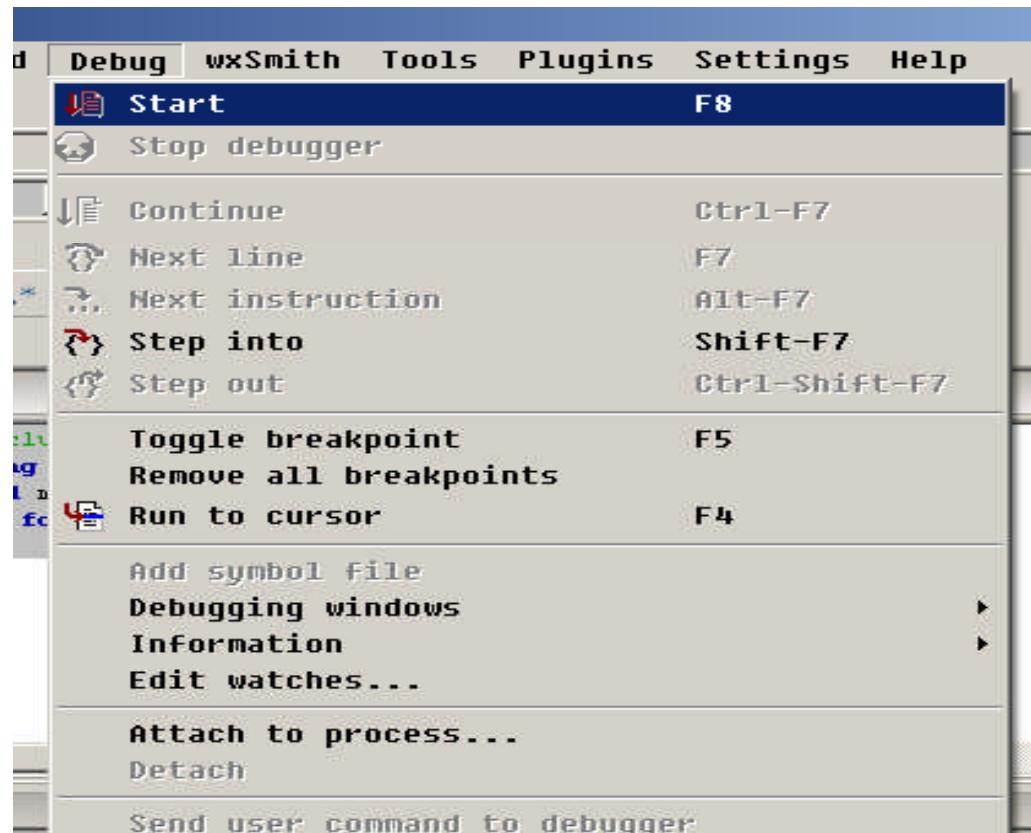


# Watch 'i'

- In ‘Edit debugger watches’,
- click on ‘Add’ and
- in the ‘add watch window’
- type ‘i’ to see the
- variable i’s value
- during debugging:

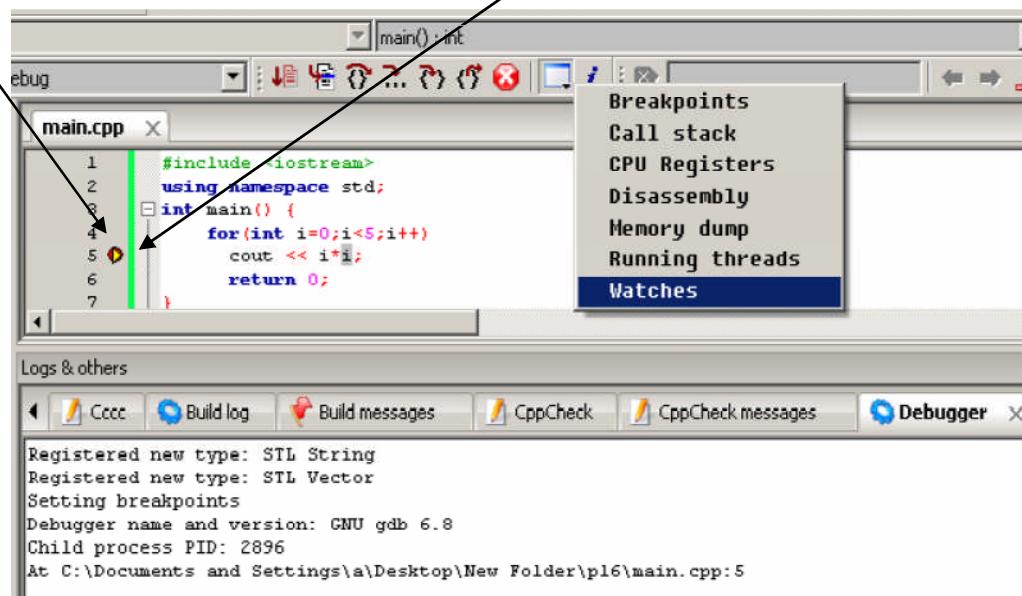


# Start debugging:



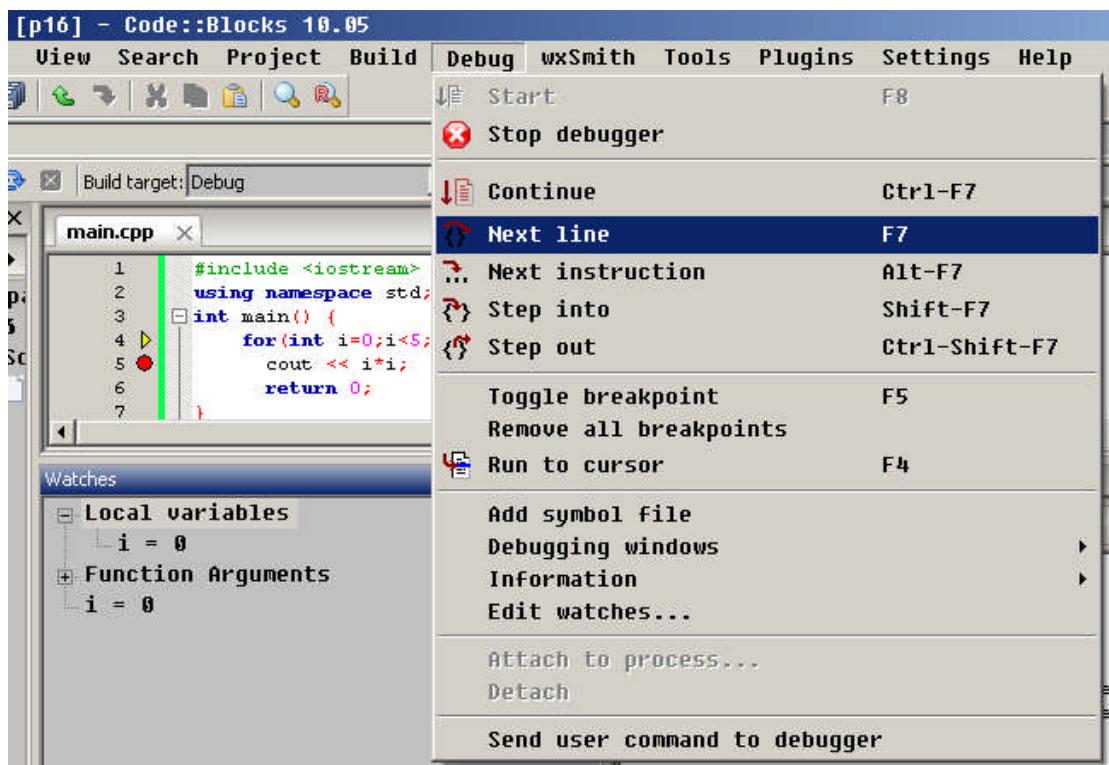
# Stopping at a breakpoint

- Now the program has stopped at the breakpoint, in the image, the red-dot has a yellow-arrow, showing that it is the current statement:



# Go line by line

- To run to ‘next line’ you can press F7 or click on the icon shown on the image:



# Step until condition

- Press Control-F7, few times till 'i=4' appears in the watch window.
- Then press F7 to till 'i=5' appears in the watch window.
- Now press Alt-F7 to run the next instruction.
- You will see the 'Disassembly' window.

The screenshot shows the Code::Blocks 10.05 IDE with the following details:

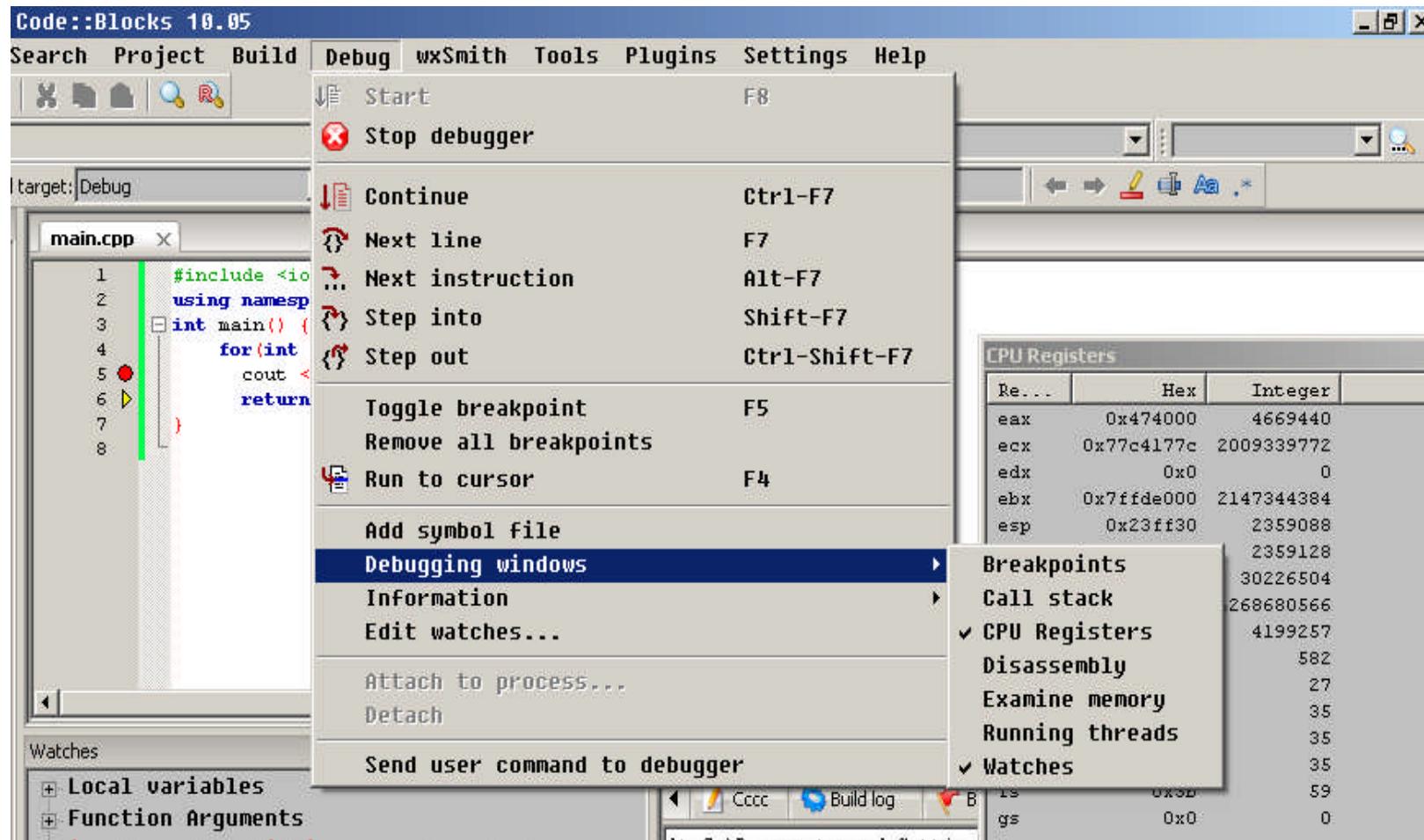
- Code::Blocks 10.05** window title.
- Search Project Build Debug wxSmith Tools Plugins Settings Help** menu bar.
- main.cpp** file open in the editor, showing the following code:

```
#include <iostream>
using namespace std;
int main() {
 for(int i=0;i<5;i++)
 cout << i*i;
 return 0;
}
```
- Watches** window:
  - Local variables: **i = 5**
  - Function Arguments: None
- Disassembly** window:
  - Function: **main (C:\Documents and Settings\...\Desktop\New)**
  - Frame start: **0023FF60**
  - Assembly code listing:

```
00401318 push %ebp
00401319 mov %esp,%ebp
0040131B and $0xffffffff,%esp
0040131E sub $0x20,%esp
00401321 call 0x413380 <__main>
00401326 movl $0x0,0x1c(%esp)
0040132E jmp 0x40134d <main+53>
00401330 mov 0x1c(%esp),%eax
00401334 imul 0x1c(%esp),%eax
00401339 mov %eax,0x4(%esp)
0040133D movl $0x4740c0,(%esp)
```

# Stop debugging

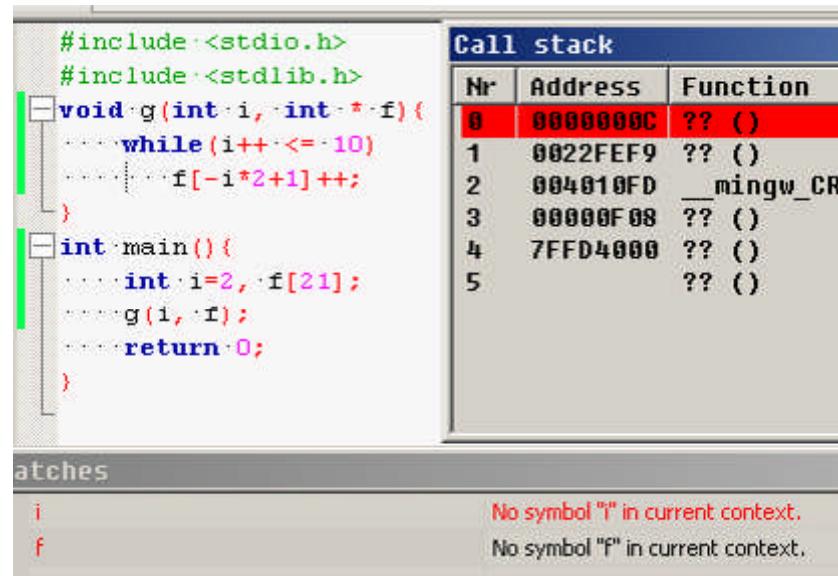
- Also look at the console window of the running program.  
To finish, click on debug > stop debugging.



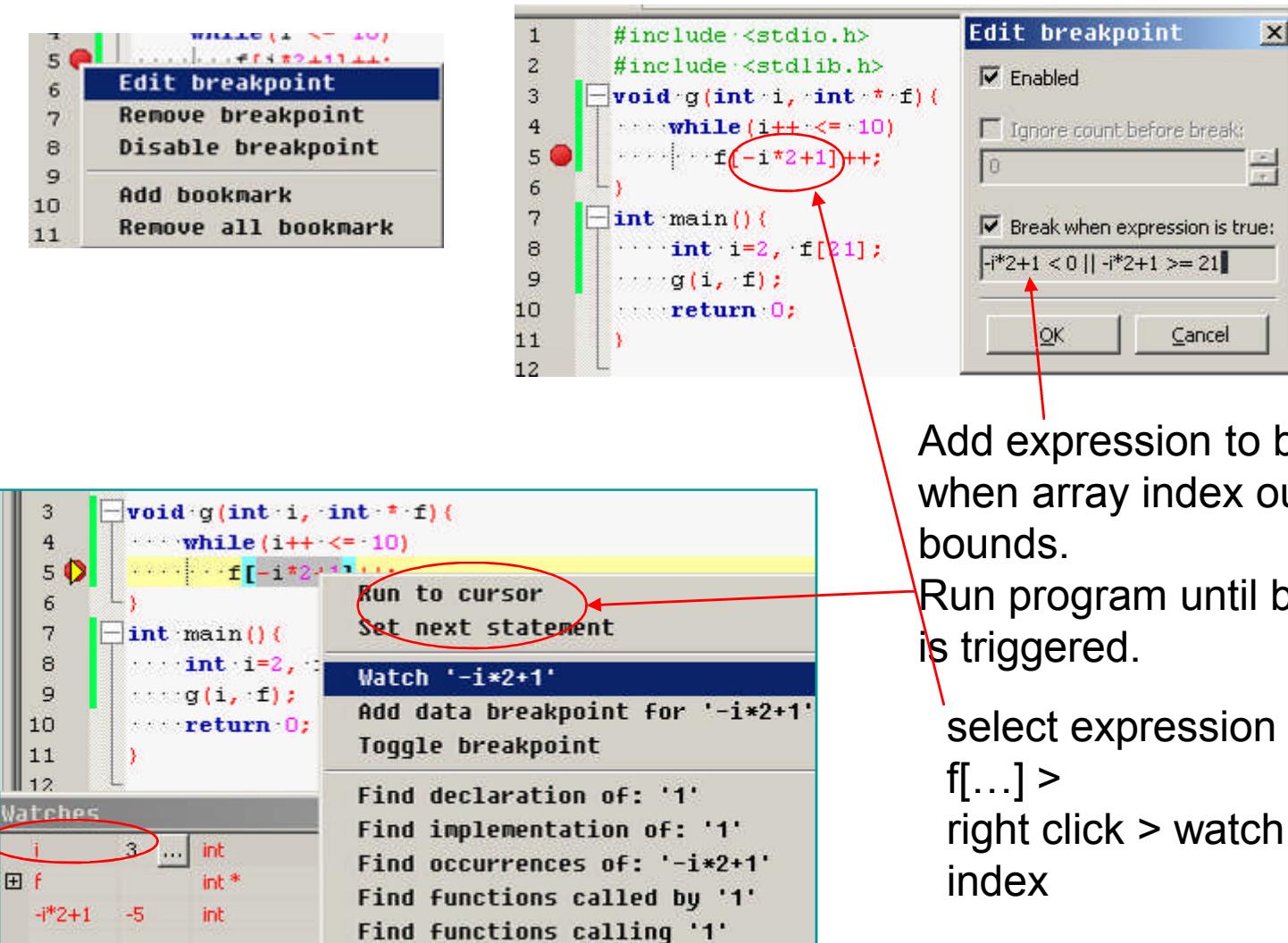
# Advanced breakpoints

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. void g(int i, int * f) {
4. while(i++ <= 10)
5. f[-i*2+1]++;
6. }
7. int main(){
8. int i=2, f[21];
9. g(i, f);
10. return 0;
11. }
```

Debug > Run > Crash



# Conditional breakpoint

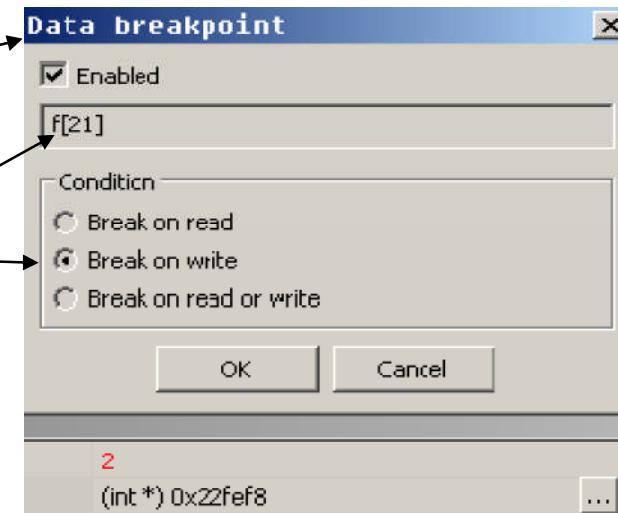


Add expression to break when array index out of bounds.  
Run program until breakpoint is triggered.  
select expression inside `f[...]` >  
right click > watch > view index

# Data breakpoint

Select f with mouse

- right click
- add data-breakpoint
- Edit data bp to 'break on write' of f[21]



# GDB commands in codeblock

If you enable gdb log window, you can see the gdb command generated by your GUI actions:

```
[debug]> break "C:/src/codeblock/bt/main.c:5"
[debug]Breakpoint 3 at 0x401339: file C:\src\codeblock\bt\main.c, line
5.
[debug]>>>>> cb_gdb:
[debug]> condition 3 -i*2+1 < 0 || -i*2+1 >= 21
[debug]>>>>> cb_gdb:
[debug]> output &f[21]
[debug](int *) 0x22ff4c>>>>>cb_gdb:
[debug]> awatch *0x22ff4c
[debug]Hardware access (read/write) watchpoint 4: *0x22ff4c
```

# Debugging tips

1. If your program crashes on some input, make a testcase:  
**hardcode** the data causing the problem, so you can debug it quickly many times (without having to type in the inputs during each debug sessions).
2. Use **binary search** to locate the cause of problems.
3. **Step over** hairy code, and watch the variables are as you expect them to be.
4. Make a **testcase**, learn how to write unittests. Run the unittest everytime you make changes to the code.
5. Use **CVS** or **GIT** to keep older versions of your program, so that you can revert to an older version of the code, if the new version doesn't work. Then use diff to find out what changes are causing the failure.
6. For really hard inconsistent problems, use **valgrind**, **purify**, and a different compiler.
7. Make notes, add comments explaining the problems.
8. Convert comments to code or assertions.
9. Search **Google** to see if others had the same problem and how they fixed it.

# Tips: Assertions and comments

```
int g(int x) {
 int t;
 // 1. t is total sum
 // 2. assume x > 0.
 ...
 // 3.
 if (t > 0)
 return 1; // true
 else
 return 0; // false

}
```

```
int g(int x) {
 int total_sum;
 assert(x>0);
 ...
 return total_sum > 0;
}
```

1. Use descriptive variables names
2. Use assert to check assumptions
3. Keep it short

# Using Microsoft MSDev VC6 IDE

(Integrated Developer Environment)

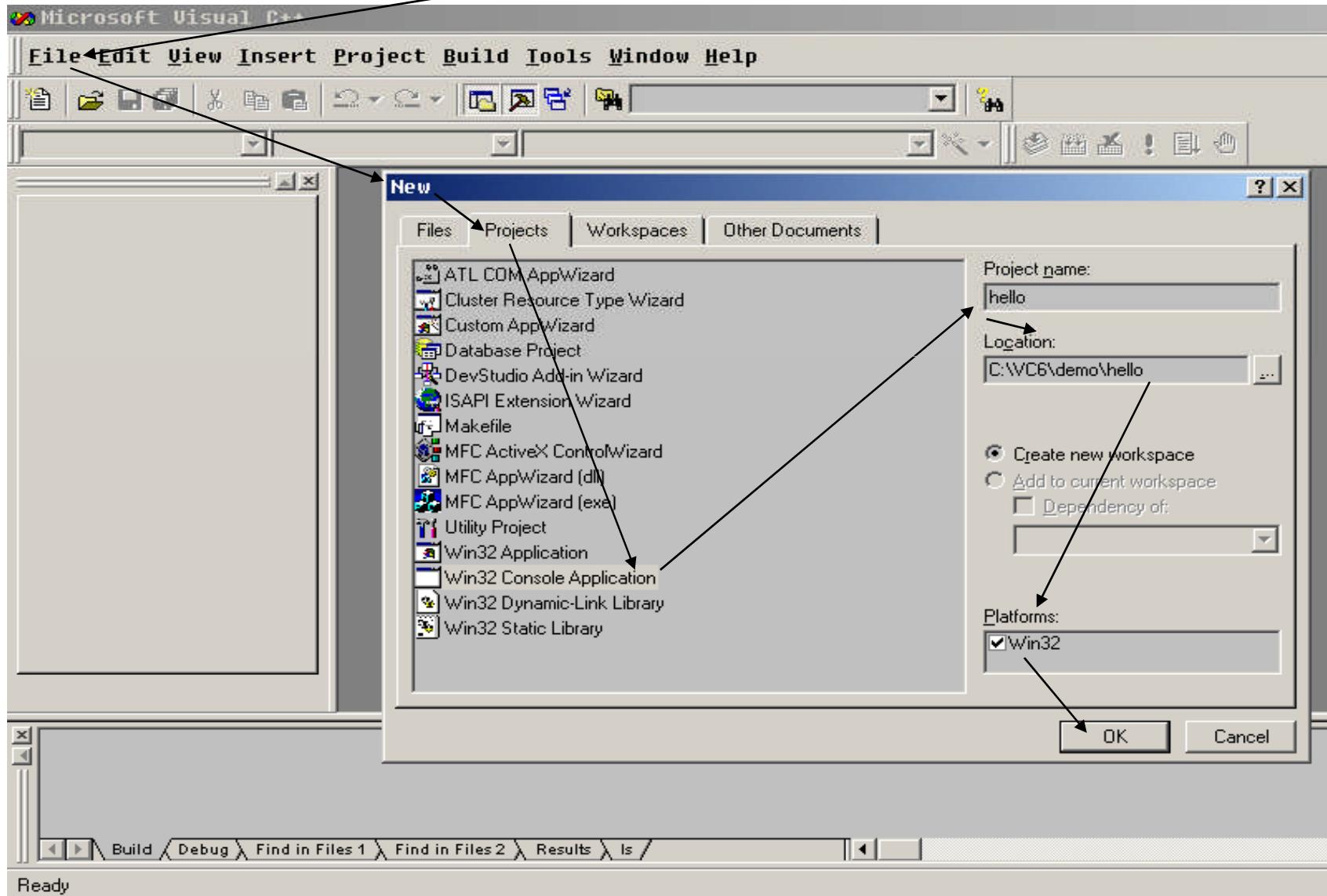


Microsoft Visual  
C++ 6.0

# Using VC6

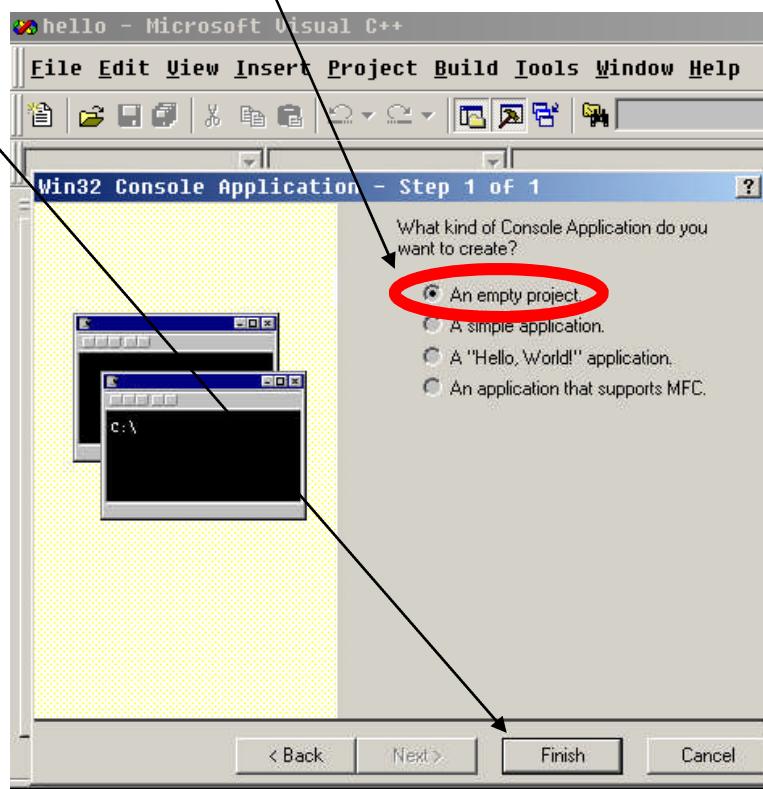
- Visual Studio from Microsoft is an IDE for C/C++ programming, msdev comes with the Microsoft C and C++ compilers and libraries and works on Windows only.
- (*Other option is to download Codeblocks with GCC (works on Linux and Windows). GCC supports C99 features, while VC++ is more useful for developing complex Windows applications.*)
- This assumes you have installed msdev vc 6.0 on your computer, with vc6 sp2 (service-pack 2 is required to compile newer C++) .
- Run msdev, then click on the msdev menu:  
**File > New > Win32 Console Application** (see next slide)
- Pick a new folder (**c:\vc6\demo\hello**) to create the application, call it hello.
- Click '**OK**' to continue the wizard.

# Creating a new application



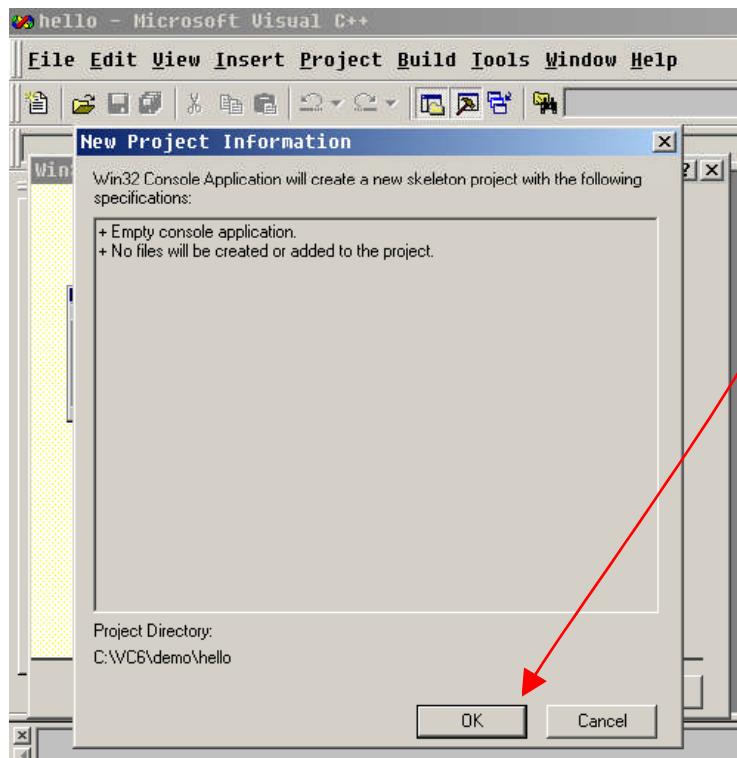
# Created an Empty project

- Select “An empty project” and
- click on “Finish”.



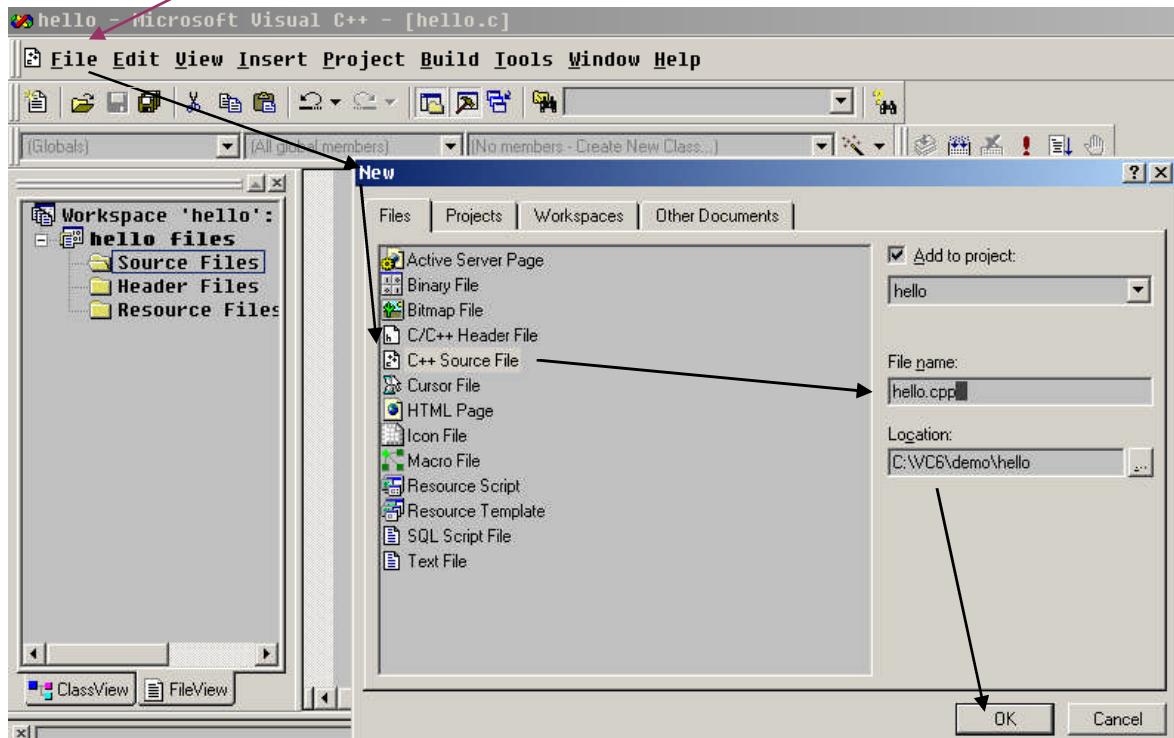
# Created a new project

- Click 'OK'



# C++ project

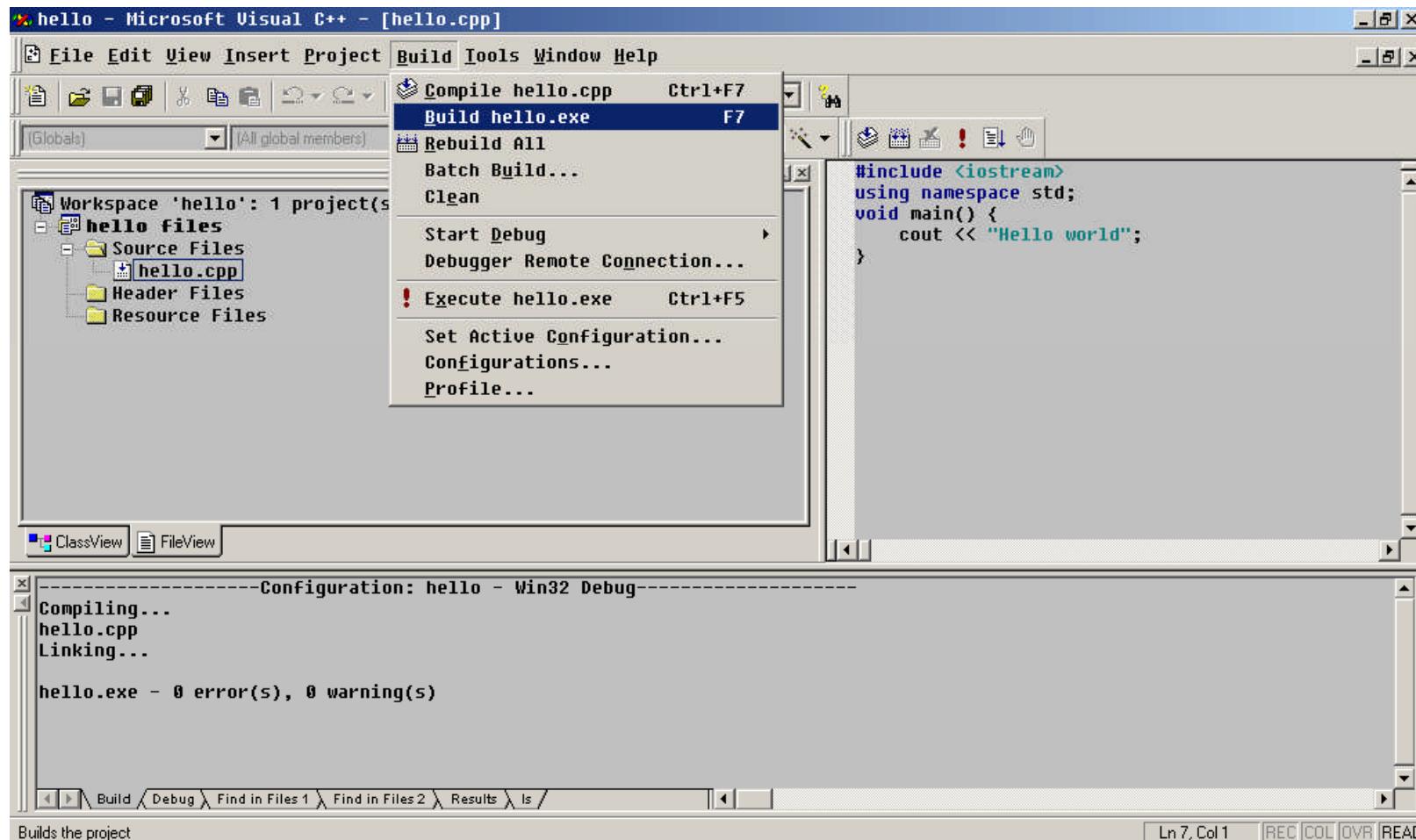
- Now click on: File > New > C++ source file > Type filename: hello.cpp



# Building the application

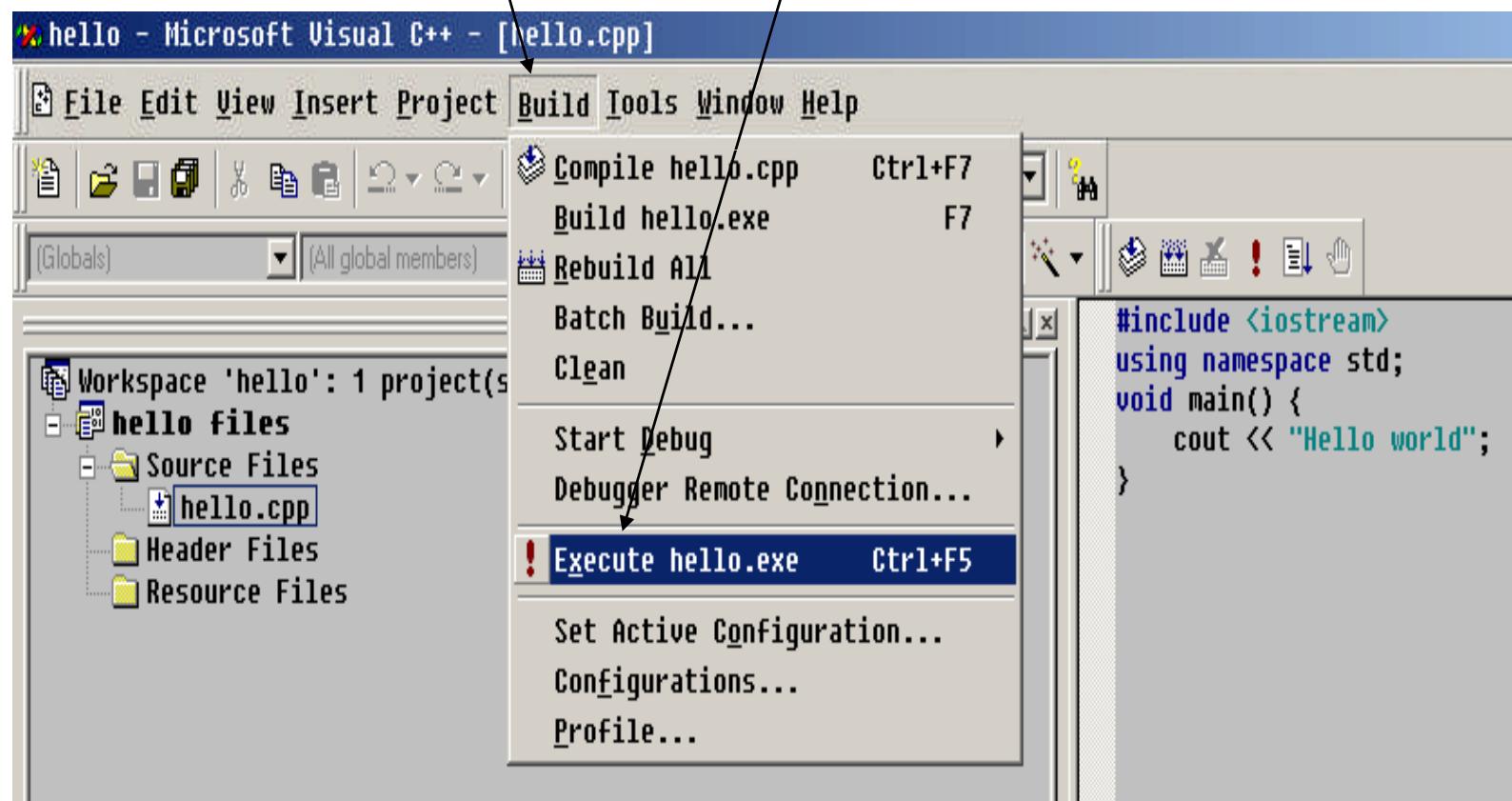
- Type the following program as hello.cpp
- Click on: **Build > Build hello.exe or [F7]**.
- In the build pane at the bottom, you will see “compiling hello.exe”.
- If there are any typos correct them
- Till you get zero errors and warnings.

# Build hello.exe



# Running the application

- Now we can run the hello-world console application: Build > Execute (C-F5)

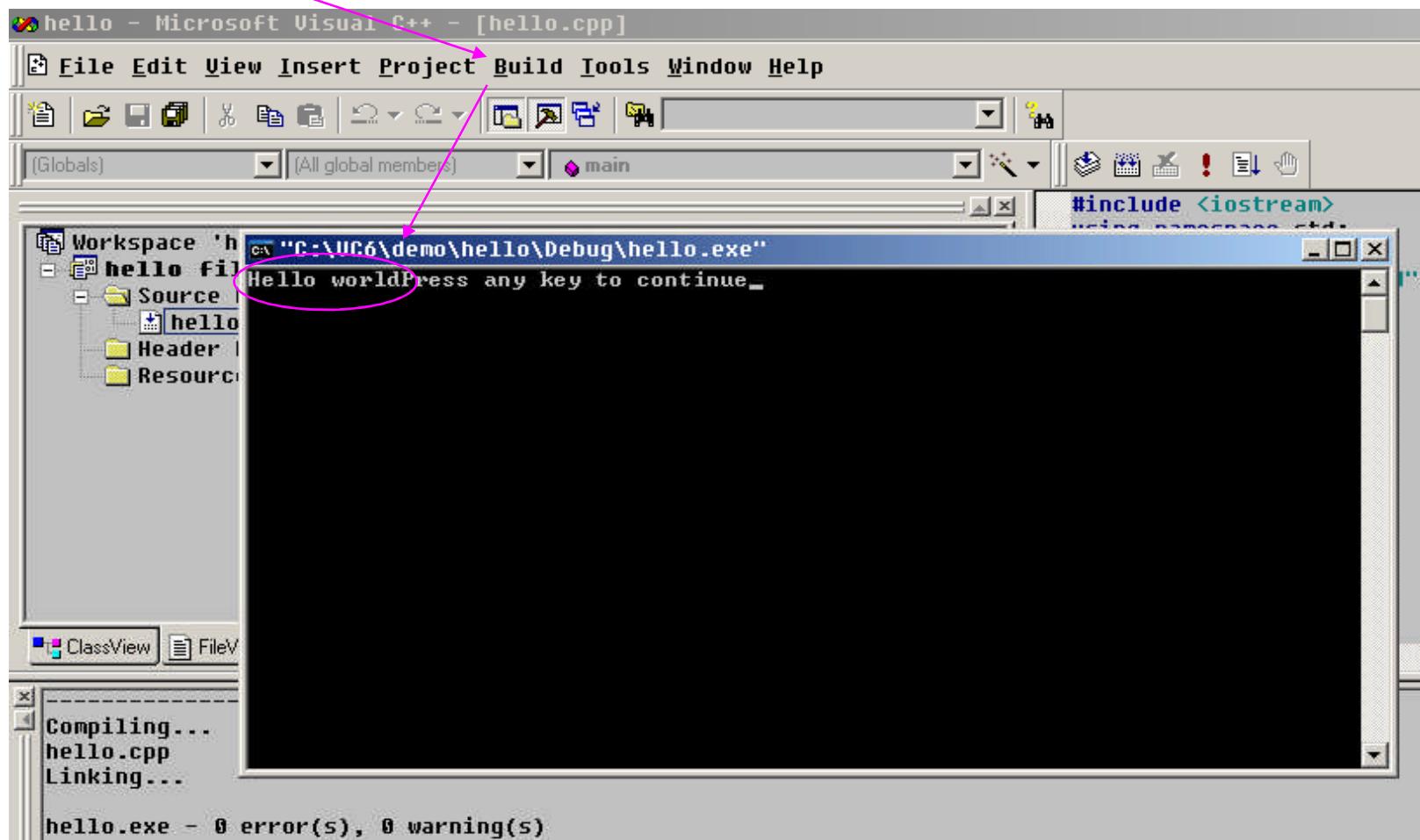


# Building until no errors

- You will see the compiler output in the build log pane on the bottom right window.
- If there are no errors, the program will run and a console window will pop up, with the output “**hello world**” (with no extra spaces around it).

# Running the application

- Build > Execute "Hello.exe"



# Debugging with MS VC6



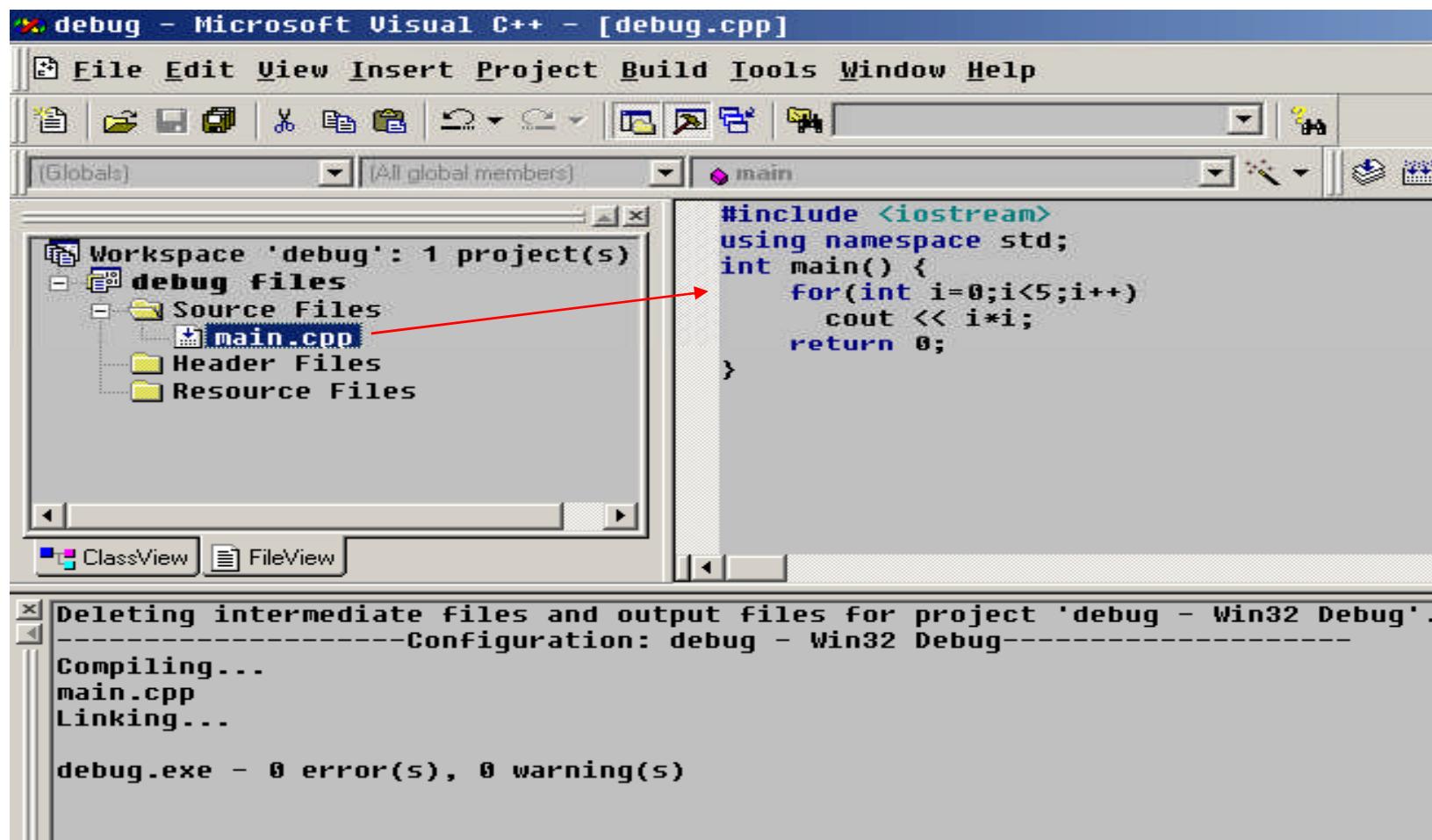
Microsoft Visual  
C++ 6.0

# Debug main.cpp

Type the following program into vc6, as main.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4 for(int i=0;i<5;i++)
5 cout << i*i;
6 return 0;
7 }
```

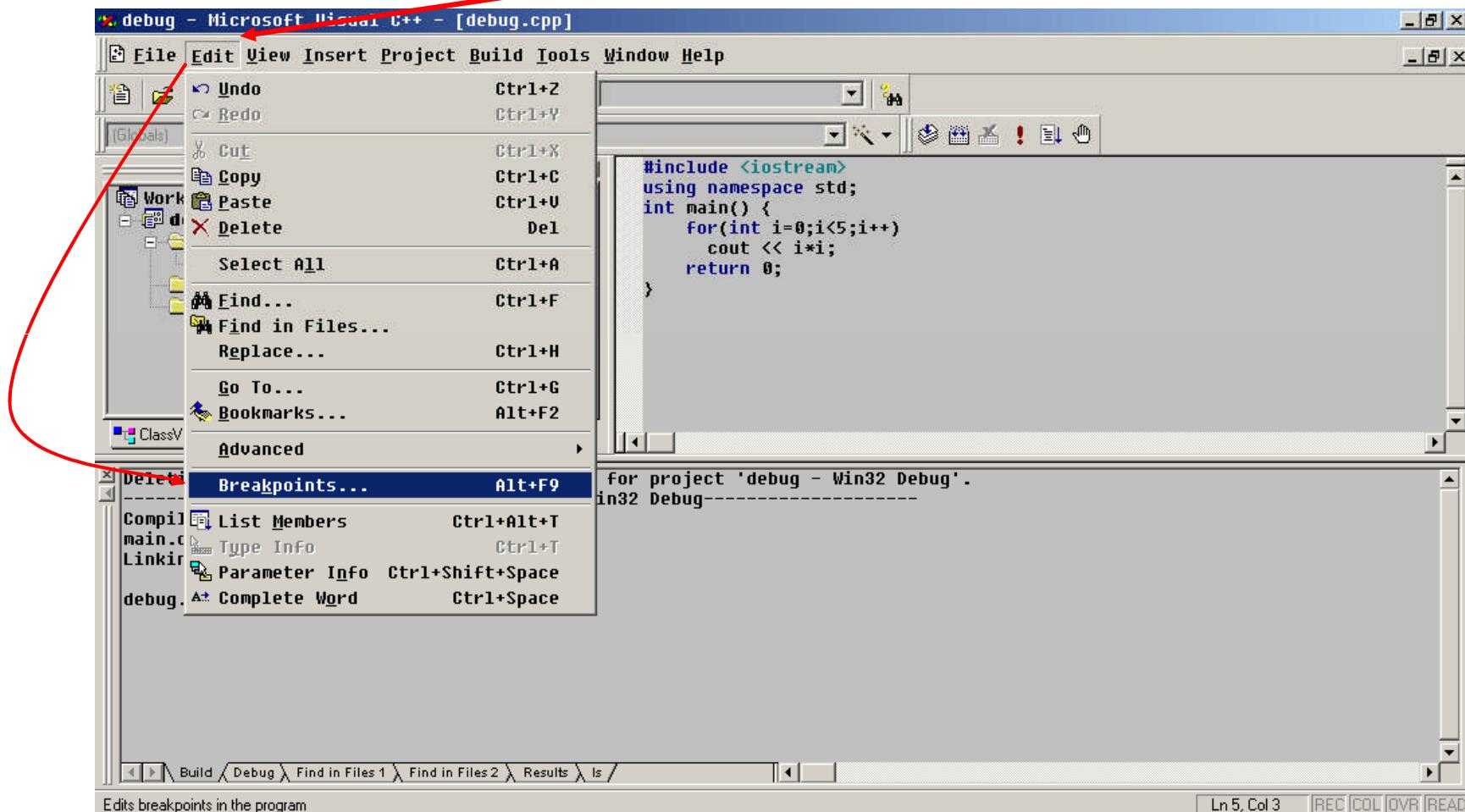
# MSDev IDE



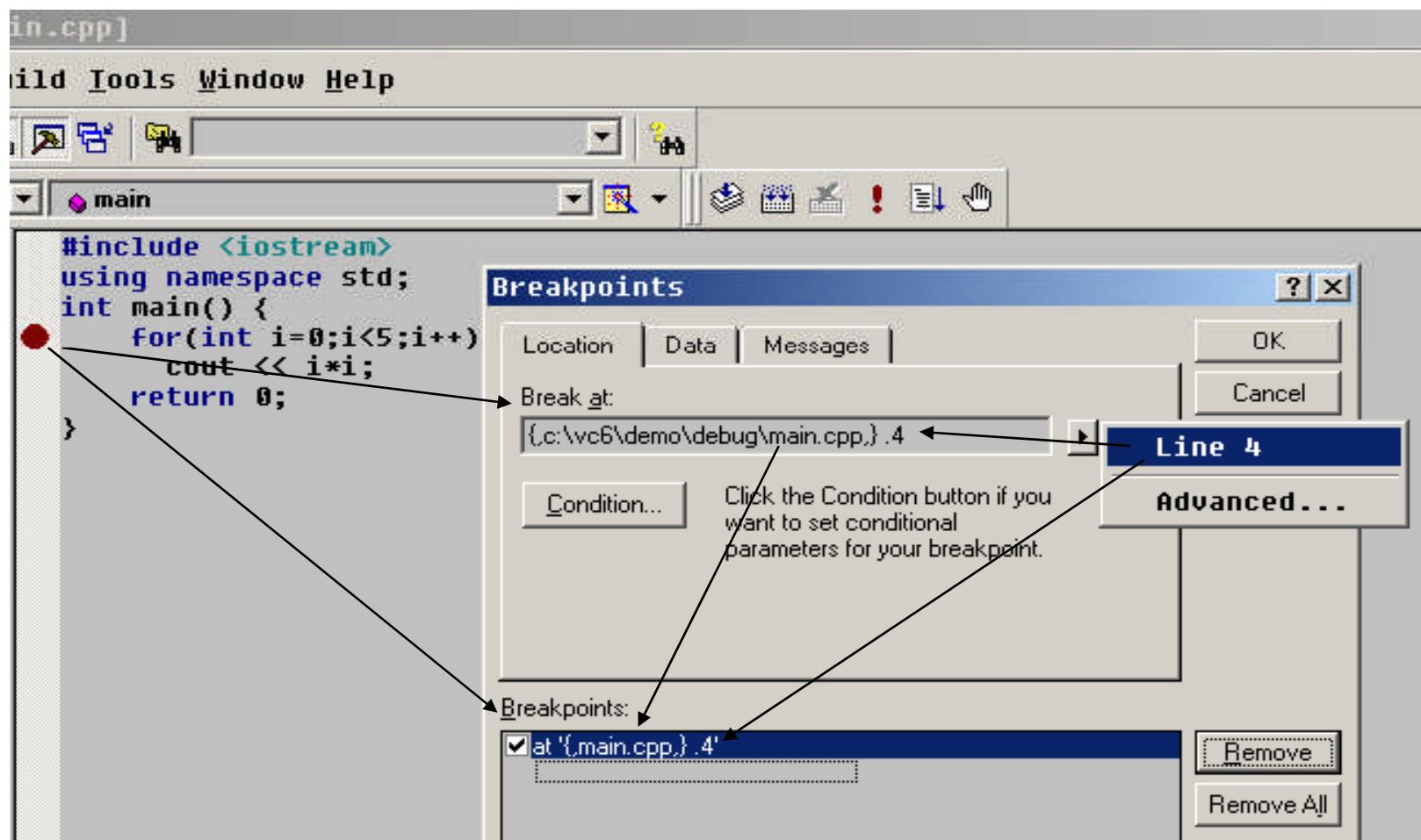
# Creating a Breakpoint

- To create a breakpoint on line 4, click on “Edit > Breakpoints...”,
- And pick ‘line 4’ from the breakpoint dialog.
- And click on ‘Build > Build main.cpp’
- And make sure it compiles with no errors.

# Accessing the breakpoints

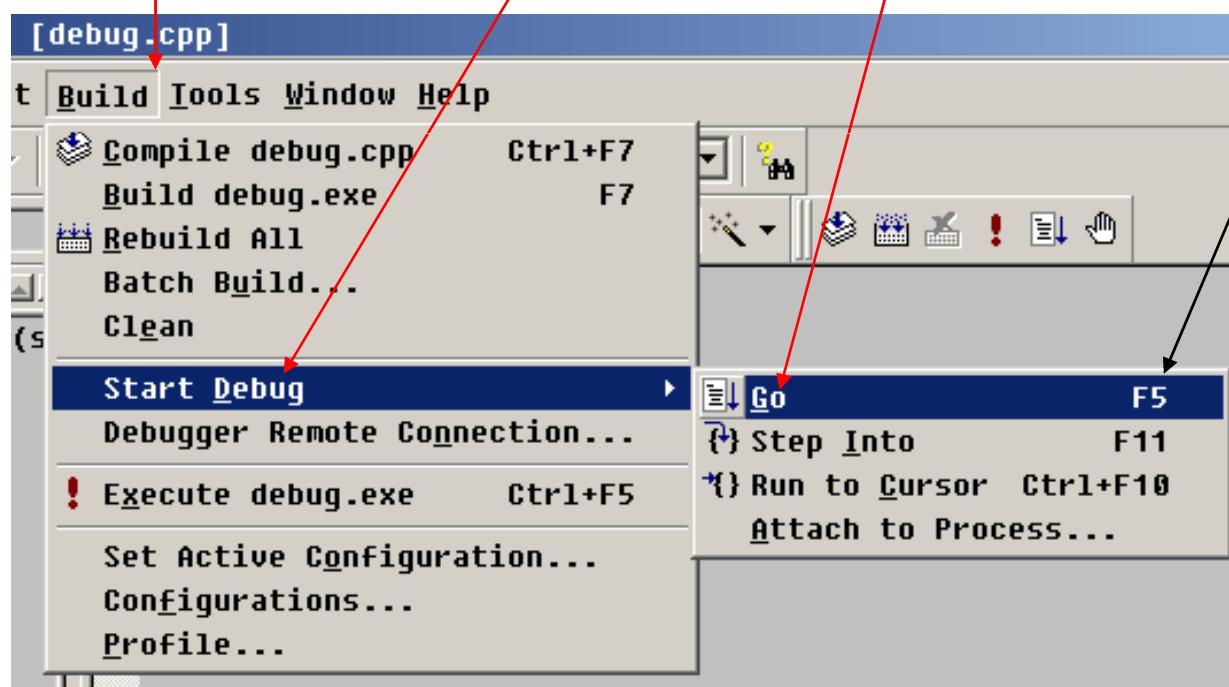


# Setting a BP



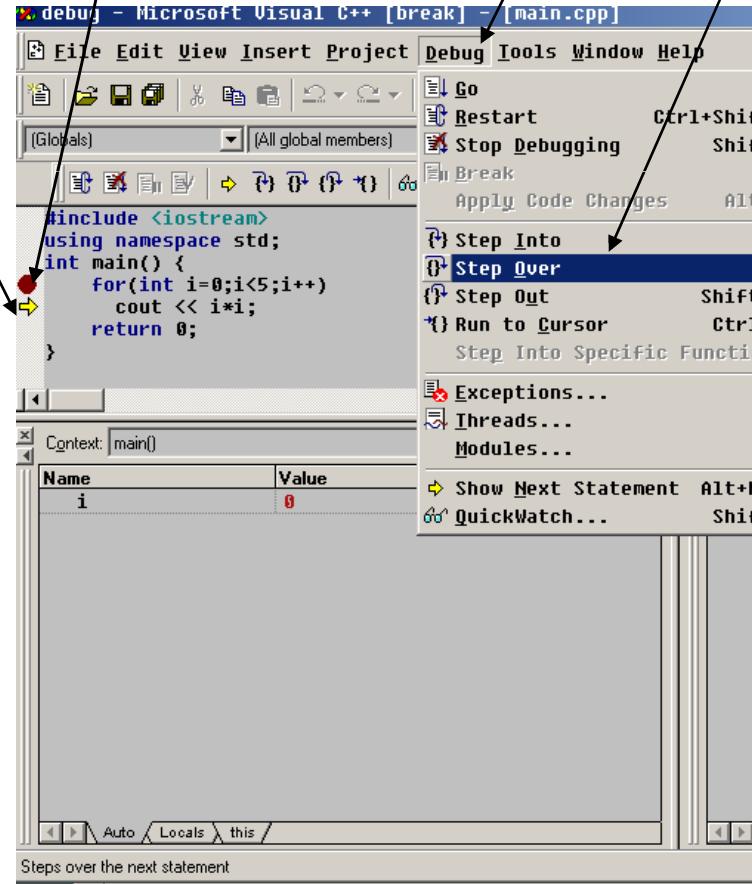
# Go, Start debugging

- Start debug by clicking on:
- Build > Start debug > Go (or press F5).



# Step by Step debugging

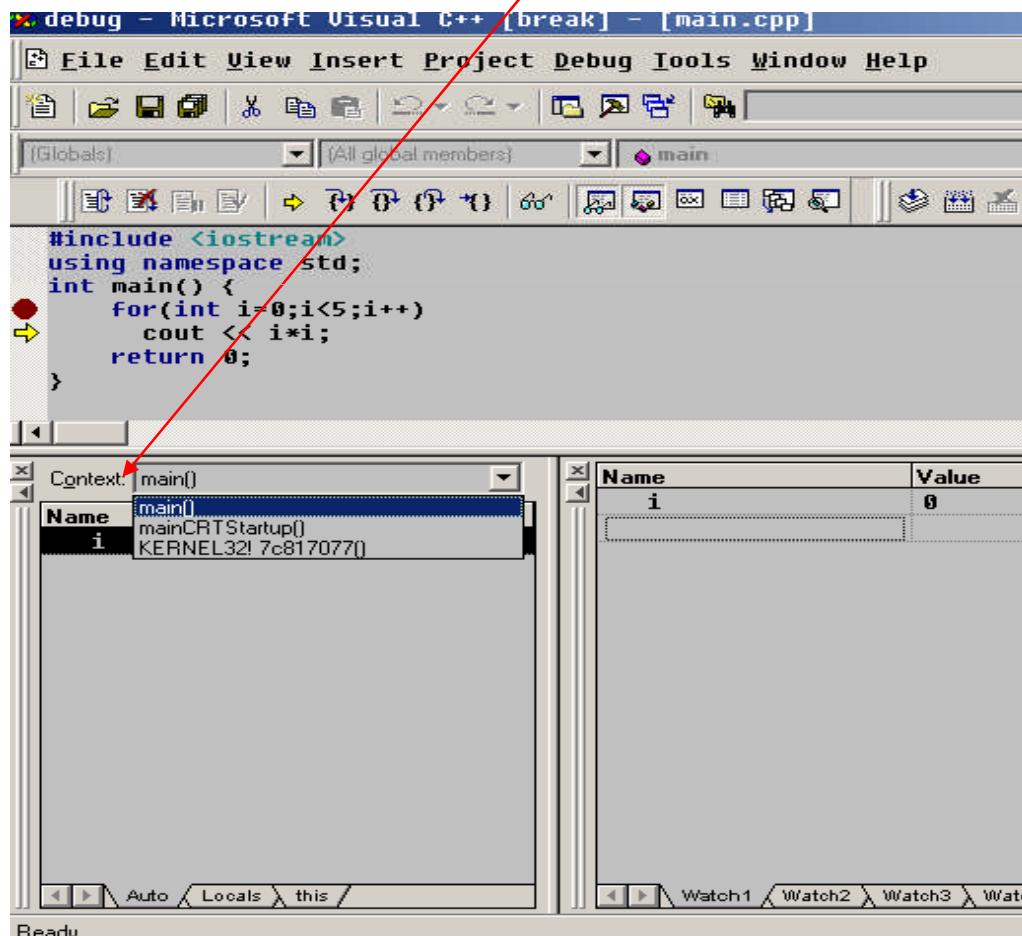
- It will stop at the breakpoint, click on: “**Debug > Step over**” to go to next line:



# Watching variables

- Below you see the value of i is ‘0’ in the ‘Auto’ pane (local variables).
- You can add any variable, e.g. ‘i’ to the watch pane, to view it’s value during debugging.
- You can also click on the ‘context’ dropdown menu to change the stackframe:

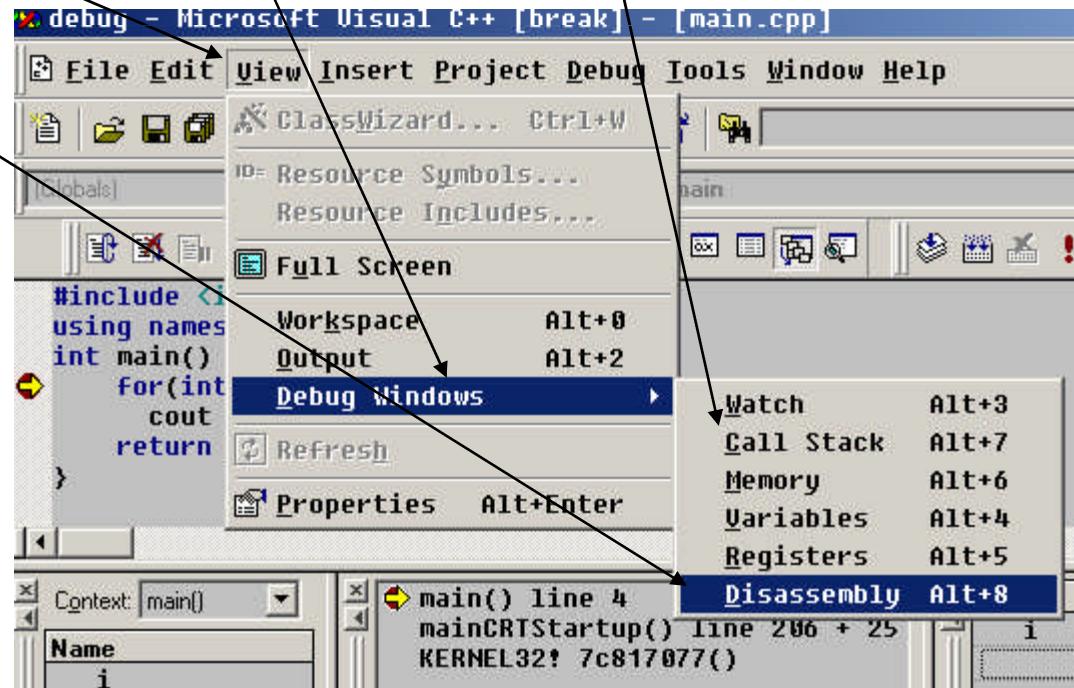
# Changing the Context (Stack frame, activation record)



# The debug windows

To view all the debug information, click on:

**View > Debug windows > call-stack or  
disassembly:**



# Dis-assembly view of source code during debugging

The screenshot shows the Microsoft Visual Studio debugger's Disassembly window. The assembly code for the `main` function is displayed, with the fourth instruction highlighted by a yellow arrow. The assembly code is as follows:

```
2: using namespace std;
3: int main() {
00401540 push ebp
00401541 mov ebp,esp
00401543 sub esp,44h
00401546 push ebx
00401547 push esi
00401548 push edi
00401549 lea edi,[ebp-44h]
0040154C mov ecx,11h
00401551 mov eax,0CCCCCCCCCh
00401556 rep stos dword ptr [edi]
4: for(int i=0;i<5;i++)
00401558 mov dword ptr [ebp-4],0
0040155F jmp main+2Ah (0040156a)
00401561 mov eax,dword ptr [ebp-4]
00401564 add eax,1
00401567 mov dword ptr [ebp-4],eax
0040156A cmp dword ptr [ebp-4],5
0040156F inc main+44h (00401584)
```

The assembly code corresponds to the following C++ source code:

```
using namespace std;
int main() {
 for(int i=0;i<5;i++)
}
```

The bottom left pane shows the `Locals` window with a variable named `i`. The bottom right pane shows the `Call Stack` window, which displays the current stack frame and the call to `KERNEL32! 7c817077()`.

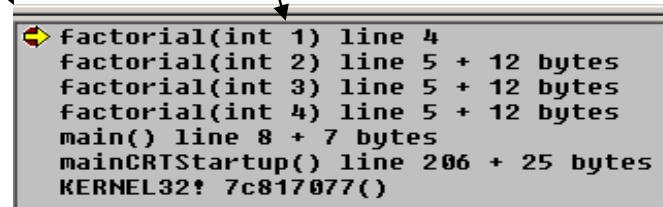
# Stack during recursion

Now type the factorial program to see the call stack during recursion.

```
1. #include <iostream>
2. using namespace std;
3. int factorial(int i){
4. if (i<2) return 1;
5. else return i * factorial(i-1);
6. }
7. int main() {
8. cout << factorial(4);
9. }
```

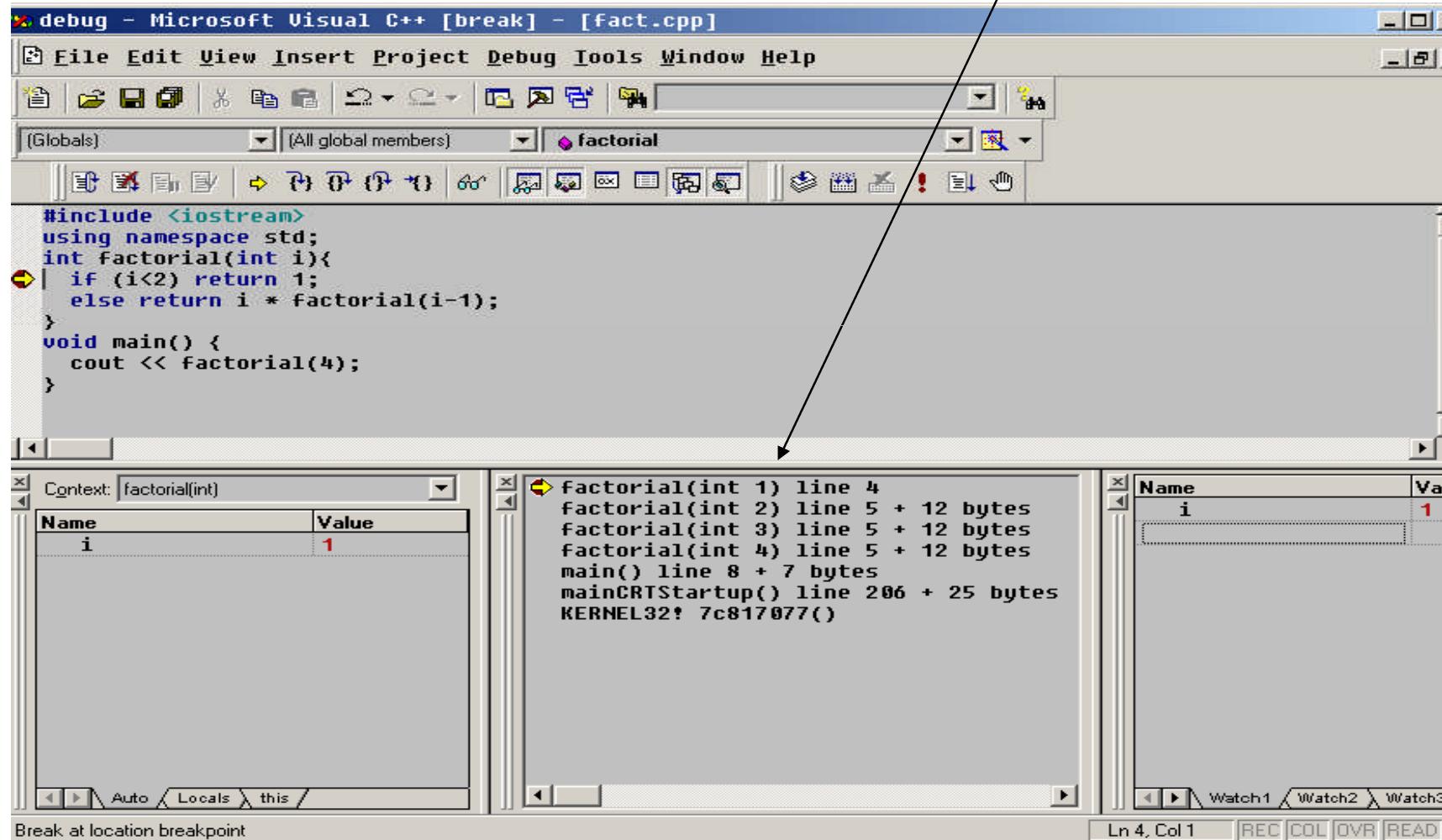
# Recursion

- Build and put a breakpoint on line 4 as shown in the image.
- Then start debugging.
- Display of ‘**call stack**’, as you step through the program,
- you will see the recursion go deeper into the factorial function until  $i=1$  is seen in the call stack.



```
Factorial(int 1) line 4
Factorial(int 2) line 5 + 12 bytes
Factorial(int 3) line 5 + 12 bytes
Factorial(int 4) line 5 + 12 bytes
main() line 8 + 7 bytes
mainCRTStartup() line 206 + 25 bytes
KERNEL32! 7c817077()
```

# Recursive call stack



# Computing 4!

- Now step ahead couple of times to see the call stack unwind, and the main function will print '24' (4! is 1\*2\*3\*4).

The screenshot shows the Microsoft Visual Studio debugger interface. The title bar reads "debug - Microsoft Visual C++ [break] - [fact.cpp]". The menu bar includes File, Edit, View, Insert, Project, Debug, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, and Build. The Globals window shows the variable "main" with its value set to "fact.cpp". The code editor displays the following C++ code:

```
#include <iostream>
using namespace std;
int Factorial(int i){
 if (i<2) return 1;
 else return i * Factorial(i-1);
}
void main() {
 cout << Factorial(4);
}
```

The cursor is at the start of the "Factorial(4)" call in the main function. The bottom pane shows the "Context: main()" tab and a table of variables:

| Name      | Value      |
|-----------|------------|
| Factorial | returne 24 |

On the right, the "Registers" window shows:

- main() line 8 + 7 bytes
- mainCRTStartup() line 206 + 25 bytes
- KERNEL32! 7c817077()

# Computed $4! = 24$ .

The screenshot shows a debugger interface with a code editor and a call stack window.

**Code Editor:**

```
#include <iostream>
using namespace std;
int Factorial(int i){
 if (i<2) return 1;
 else return i * Factorial(i-1);
}
void main() {
 cout << Factorial(4);
}
```

**Call Stack:**

Context: main()

| Name                    | Value                             |
|-------------------------|-----------------------------------|
| std::basic_ostream<...> | C:\VC6\demo\debug\Debug\debug.exe |

**Output Window:**

```
24
```

# Hardware breakpoints

Visual C++ only supports **write-only data breakpoints**,  
example use to find out "*Who is corrupting our data structure?*"  
Consider this program, why is it printing k=202, i=301 (not k=201)?

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. //
4. void update(int mr[]) {
5. int i;
6. for(i=0;i<=3;i++)
7. mr[i]++;
8. }
9. int main() {
10. int k=201, ar[3] = {10,20,30}, i=301;
11. update(ar);
12. printf("k=%d, i=%d",k,i);
13. return 0;
14. }
```

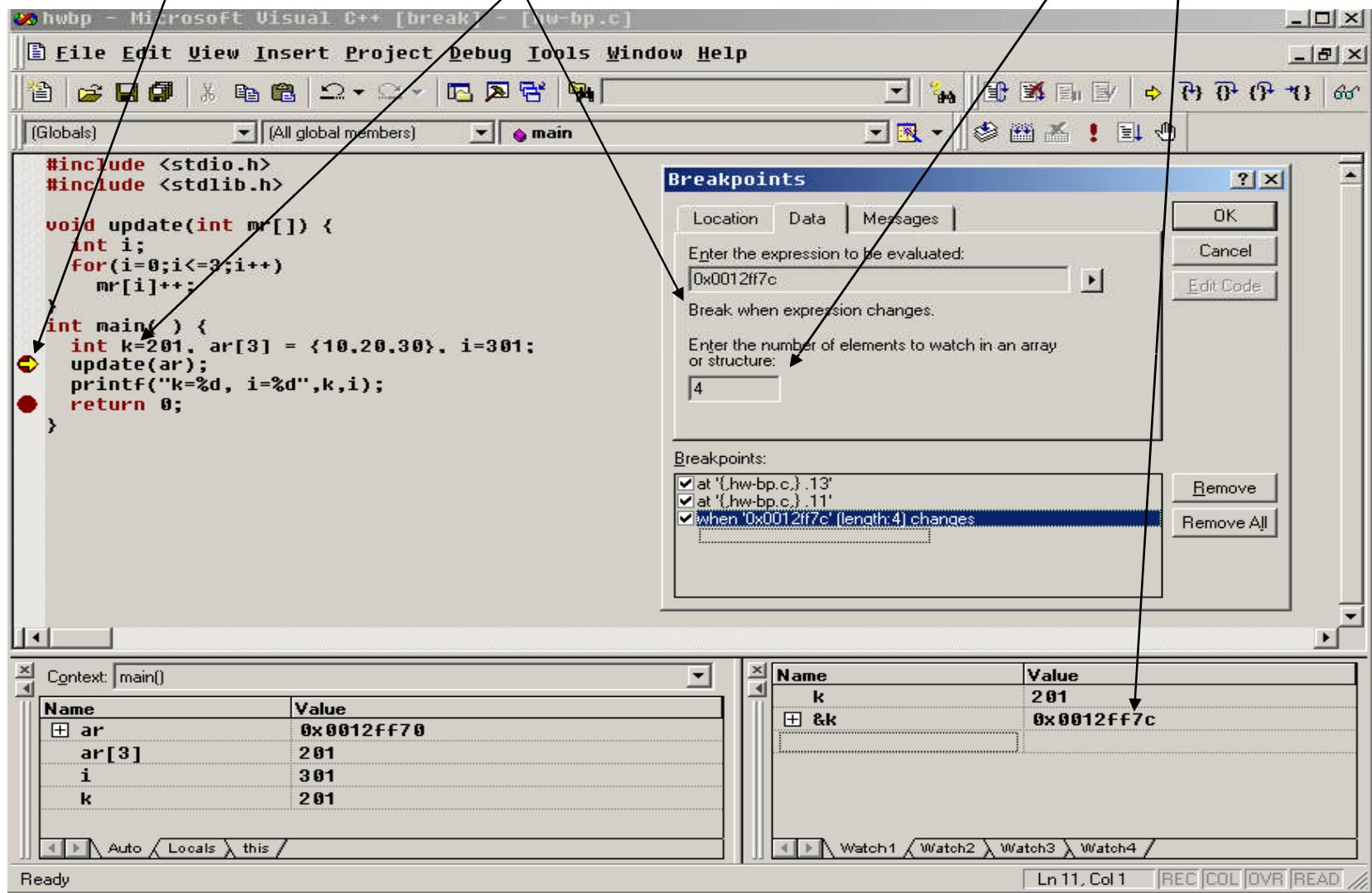
# Debug registers

- The Intel x86 CPUs have some special registers which are intended for debugging use only. By storing special values into these registers, a program can ask the CPU to execute an INT 1 (interrupt 1) instruction immediately whenever a specified memory location is read from or written to.
- **INT 1** also happens to be the interrupt that's executed by the CPU after a debugger asks the CPU to single-step one assembly line of the program.

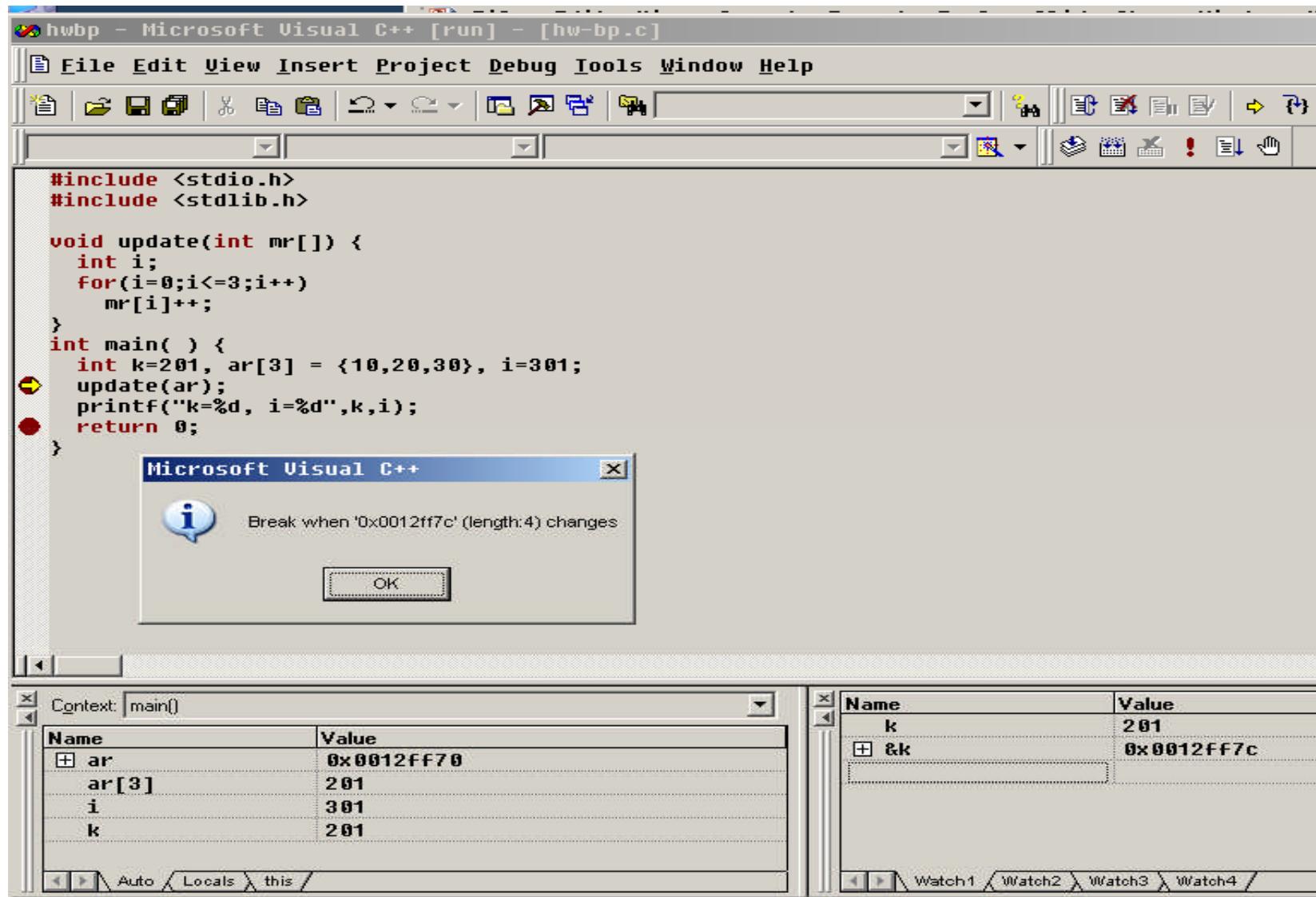
# Watching memory for change

- Determine what address you want to watch
- Open the new **breakpoint** dialog, and switch to the data tab
- Enter the address that you want to watch in the variable column
- The context is ignored. You can clear this if you want.
- Set the item count. If you are using an address, this should be the number of bytes (example: 4 for a integer type)

Stop after k is allocated, find its address, and put a **Data change breakpoint** on 4 bytes of k



## Breakpoint hit in update(), when k is changed



At the bp i=3 and  $mr[3]++ = k++$

The screenshot shows the Microsoft Visual Studio C++ debugger interface. The title bar reads "hwbp - Microsoft Visual C++ [break] - [hw-bp.c]". The menu bar includes File, Edit, View, Insert, Project, Debug, Tools, Window, and Help. The toolbar has various icons for file operations and debugging. The code editor window displays the following C code:

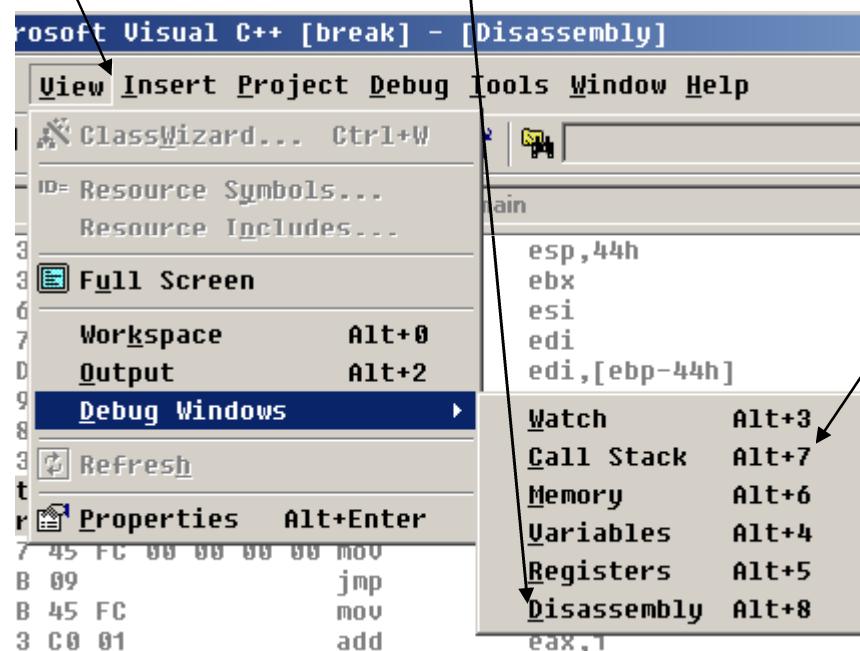
```
#include <stdio.h>
#include <stdlib.h>

void update(int mr[]) {
 int i;
 for(i=0;i<=3;i++)
 mr[i]++;
}

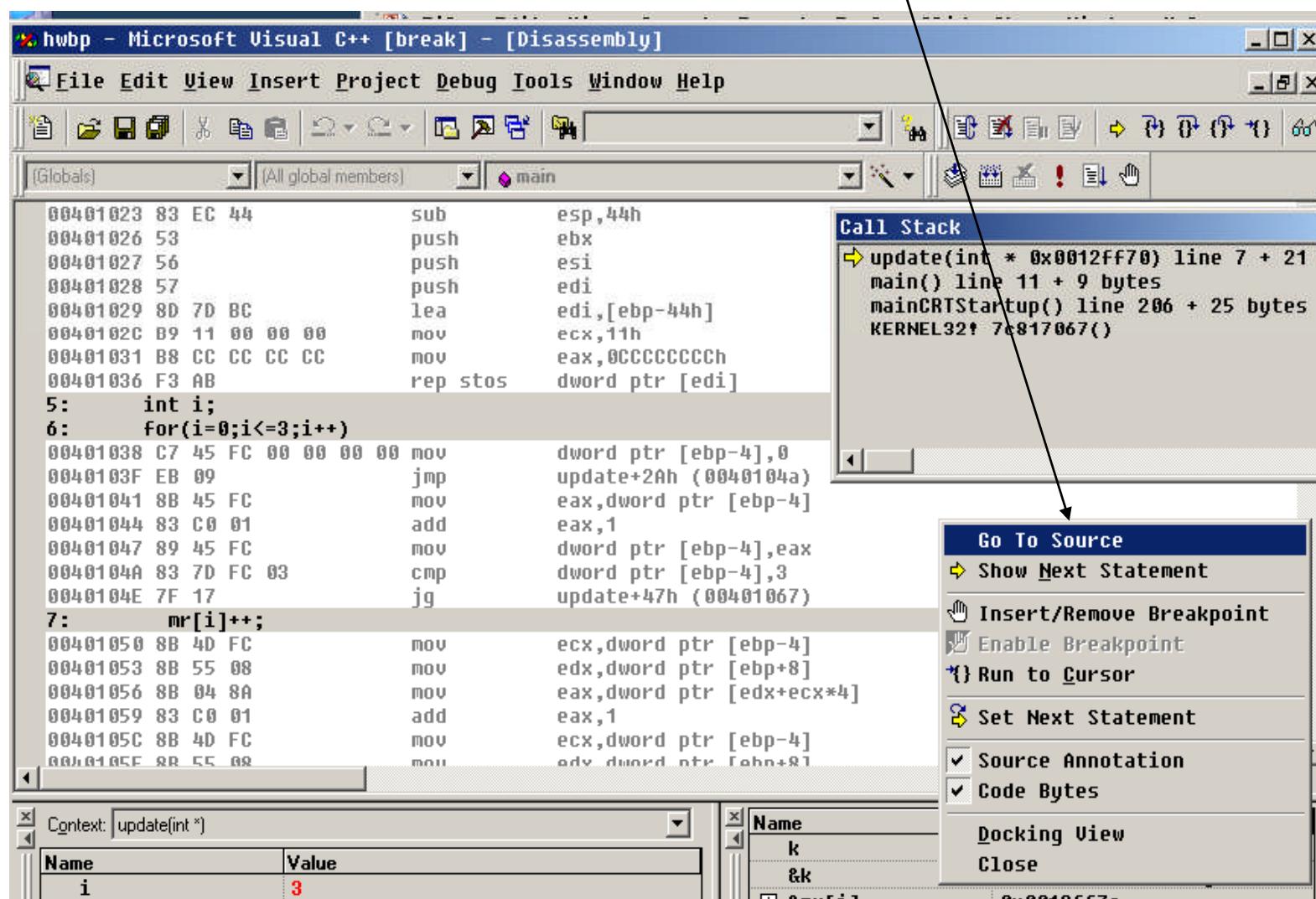
int main() {
 int k=201, ar[3] = {10,20,30}, i=301;
 update(ar);
 printf("k=%d, i=%d",k,i);
 return 0;
}
```

A yellow arrow points to the first line of the update function, indicating it is the current instruction. Three red circles mark the breakpoints at the start of the update function, before the call to update, and before the return statement. The Locals window shows the variable context for the update function, with **i** set to 3 and **mr[i]** set to 202. The Watch window shows the values for **k** and **&k**, both of which are marked with the error message "CXX0017: Error: symbol 'k'".

# View the disassembly and call stack

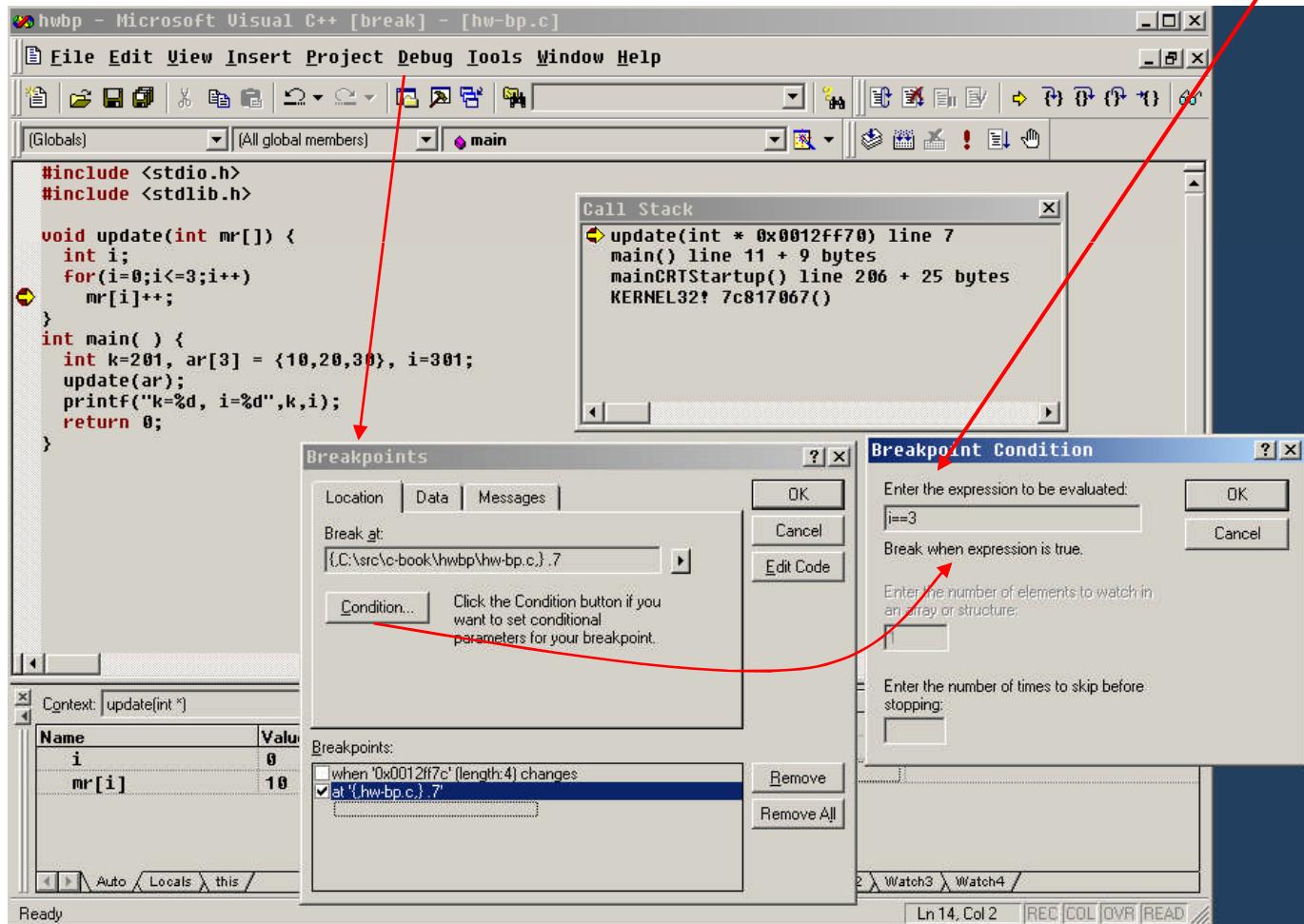


# Right click to see the source

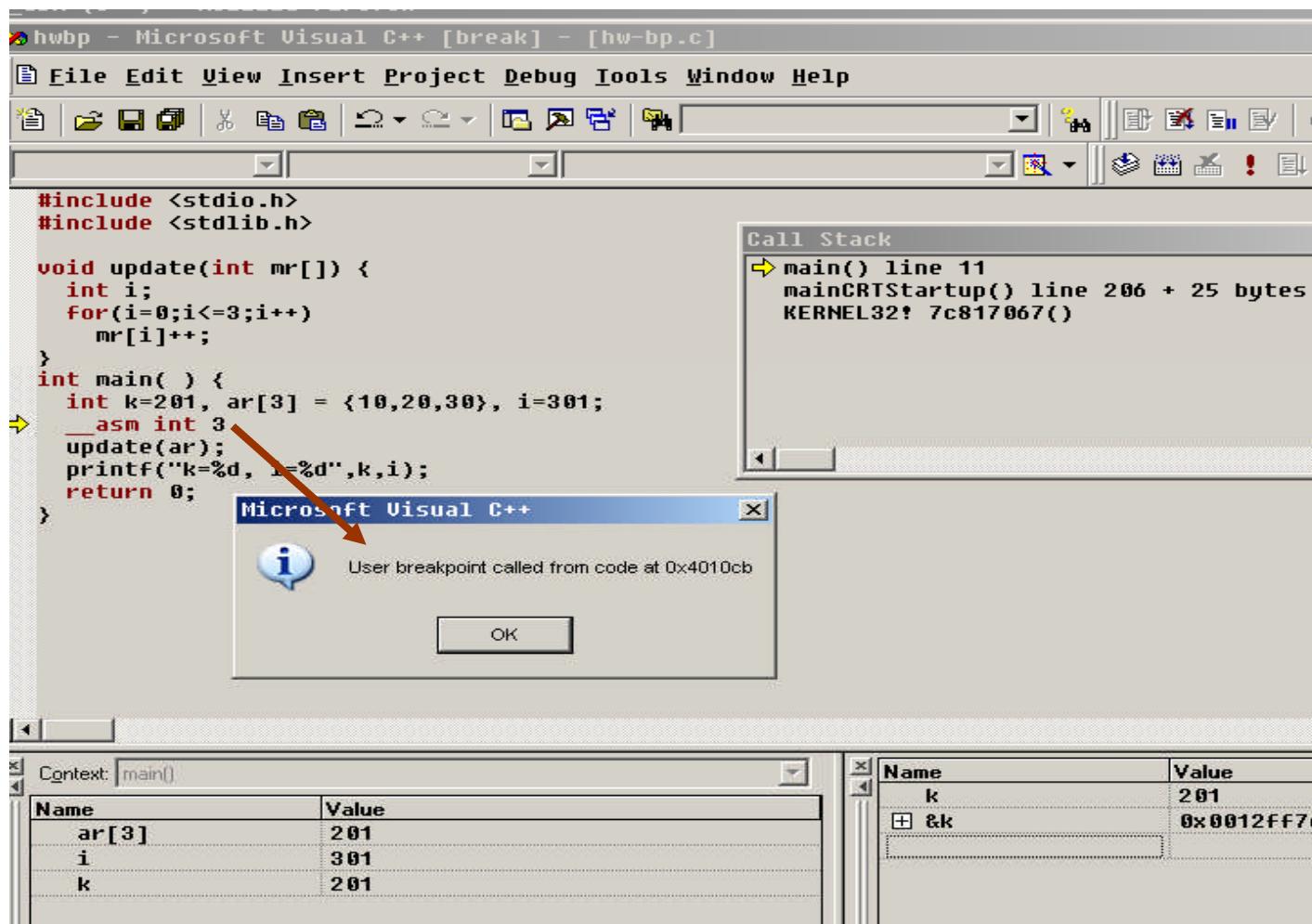


# Conditional Breakpoint

- We want to stop in the loop, only when  $i=3$



# Hardcoded Breakpoint: \_\_asm int 3



# VC Command line setup

After Installing VC6 in c:\vc6,

**Start > Run > Cmd**

```
C:\> c:\vc6\VC98\bin\VCVARS32.BAT
set VSCommonDir=C:\vc6\common
set MSDevDir=C:\vc6\common\msdev98
set MSVCDir=C:\vc6\VC98
set PATH=%MSDevDir%\BIN;%MSVCDir%\BIN;
 %VSCommonDir%\TOOLS\WINNT;
 %VSCommonDir%\TOOLS;
 %PATH%
set INCLUDE=%MSVCDir%\ATL\INCLUDE;
 %MSVCDir%\INCLUDE;
 %MSVCDir%\MFC\INCLUDE;
 %INCLUDE%
set LIB=%MSVCDir%\LIB;
 %MSVCDir%\MFC\LIB;
 %LIB%
```

# VC Command line compiler

```
C:\> cl /W4 /ZI /GZ hello.c .. To compile debug hello.exe
C:\> cl /W4 hello.c | gvim -q - .. Quick-Fix errors with vim
```

```

C:\> cl /? .. help with c compiler
```

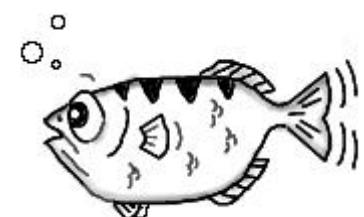
**Useful Flags:**

- /GX .. enable C++ Exception Handler
- /W4 .. all warnings
- /ZI .. incremental debugger
- /GZ .. runtime debug checks

```
C:\> ntsd .. command line debugger (if gcc doesn't work)
```

# Debugging with

# GDB



# Compiling for gdb

On MS-Windows, install codeblocks or cygwin:

```
C:\> set path=c:\codeblocks\mingw\bin;%path%
```

Linux should have gdb pre-installed.

Must compile with gcc '**-g**' debug flag  
and **-Wall** to see all warnings:

```
$ gcc -g -Wall test1.c -o test1.exe
```

```
$ g++ -g -Wall test2.cpp -o test2.exe
```

# Using gdb

```
$ gdb test1.exe
```

```
(gdb) help # to see all the commands
```

```
(gdb) run [args to test1.exe]
```

```
Hello world
```

```
(gdb) kill # if program is not done
```

```
(gdb) quit
```

```
$
```

# Stack

- (gdb) **bt** # backtrace
  - #0 func2 (x=30) at test.c:5
  - #1 0xe6 in func1 (a=30) at test.c:10
  - #2 0x80 in main (argc=1, argv=0xbff) at test.c:19
  - #3 0xf23 in \_\_libc\_start\_main () from /lib/libc.so.6
- (gdb) **frame 2**
  - #2 0x814 in main (argc=1, argv=0xf4) at test.c:19 ..

# What's on the stack

- (gdb) **info frame**
  - Stack level 2, frame at 0xbffffa8c:
  - eip = 0x8048414 in main (test.c:19);
  - saved eip 0x40037f5c
    - called by frame at 0xbffffac8,
    - caller of frame at 0xbffffa5c source language c.
  - Arglist at 0xbffffa8c, args: argc=1, argv=0xbffffaf4
  - Locals at 0xbffffa8c,
  - Previous frame's sp is 0x0
  - Saved registers: ebp at 0xbffffa8c, eip at 0xbffffa90
- (gdb) **info locals**
  - x = 30
  - s = 0x8048484 "Hello World!\n"
- (gdb) **info args**
  - argc = 1
  - argv = (char \*\*) 0xbffffaf4

# Breakpoints (bp)

- (gdb) **break** test.c:19 # break filename:linenumber
  - Breakpoint 2 at 0x80483f8: file test.c, line 19
- (gdb) **break** func1 # break functionName
  - Breakpoint 3 at 0x80483ca: file test.c, line 10
- (gdb) **break** TestClass::testFunc(int)
  - Breakpoint 1 at 0x80485b2: file cpptest.cpp, line 16.
- (gdb) **tbreak** main # Temp breakpoint
  - Will stop once in main
- (gdb) **info breakpoints**
- (gdb) **disable** 2 # turn off bp
- (gdb) **enable** 2 # turn on bp
- (gdb) **ignore** 2 5 # skip 5 times, bp 2
  - Will ignore next 5 crossings of breakpoint 2.

# Running

- (gdb) **run**
- You Press <Control-C>
  - Program received signal SIGINT, Interrupt.
- (gdb) **bt** # backtrace, show call stack
- (gdb) **list** # show near source code.
- (gdb) **print** x # show variable value

# Stepping

- (gdb) **call** your\_c\_function(3)
- (gdb) **next** # go line by line of source code.
  - (gdb) **step** # go **into** function also.
  - (gdb) **finish** # step **out** of function
- (gdb) **continue**

# Viewing formatted data, **x/format**

(gdb) **x/s** ptr # print var as a string.

(gdb) **x/4c** ptr # as 4 chars.

(gdb) **x/t** ptr # in binary bits

(gdb) **x/3x** ptr # as 3 hex bytes

(gdb) **set var** ptr = 0 # change var

(gdb) **info** registers

# Watching data with hardware breakpoints

(gdb) **watch** x

Hardware watchpoint 4: x,  
will print message whenever x is changed.

(gdb) **rwatch** y # read bp

(gdb) **awatch** z # access.

# Viewing asm code

```
(gdb) dis main # disassemble exe
```

```
C:\> gcc -S file.c # Generate asm file
```

```
C:\> cat file.S # view it
```

...

# gdb with gui

- C:\> **gdb –tui ..** START gdb with GUI
- **Codeblocks** debugger has a gdb command line (windows and linux)
- **Emacs:** **M-x gdb** (For linux and emacs)

# Also see

- **gdb** – most systems. `~/.gdbinit`
- MS Windows:
  - **Visual studio**
  - kernel – **windbg**, softice
- Linux kernel - **kgdb**, **kdb**
- GUI – **Codeblocks**, **ddd**
- **Valgrind**, **purify** – Runtime memory errors
- **Lint** – Static, bad style warnings;
- **gcc** –**Wall** .. Warning.

# C Program

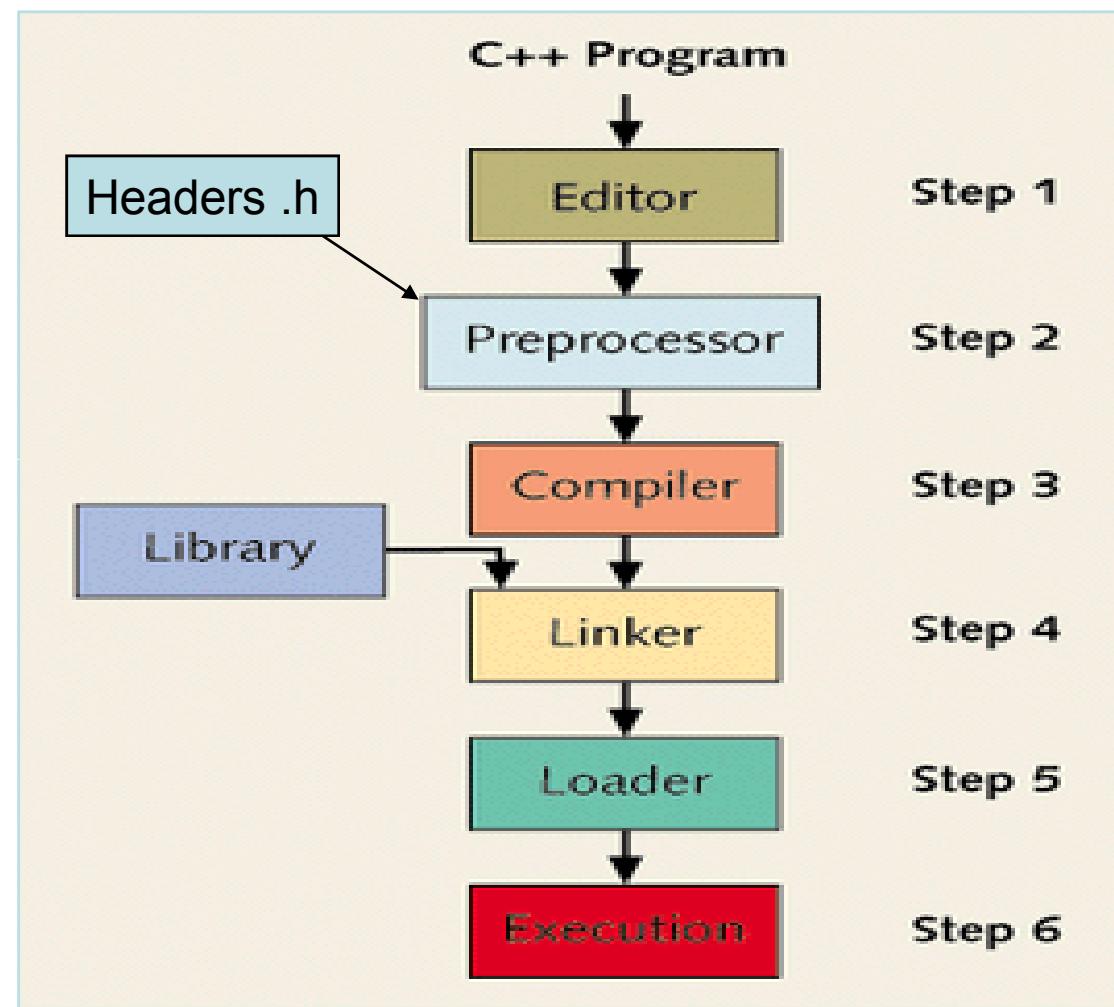
## Source to

# Execution

- Stack
- Heap memory
- Operation system

# Compiler

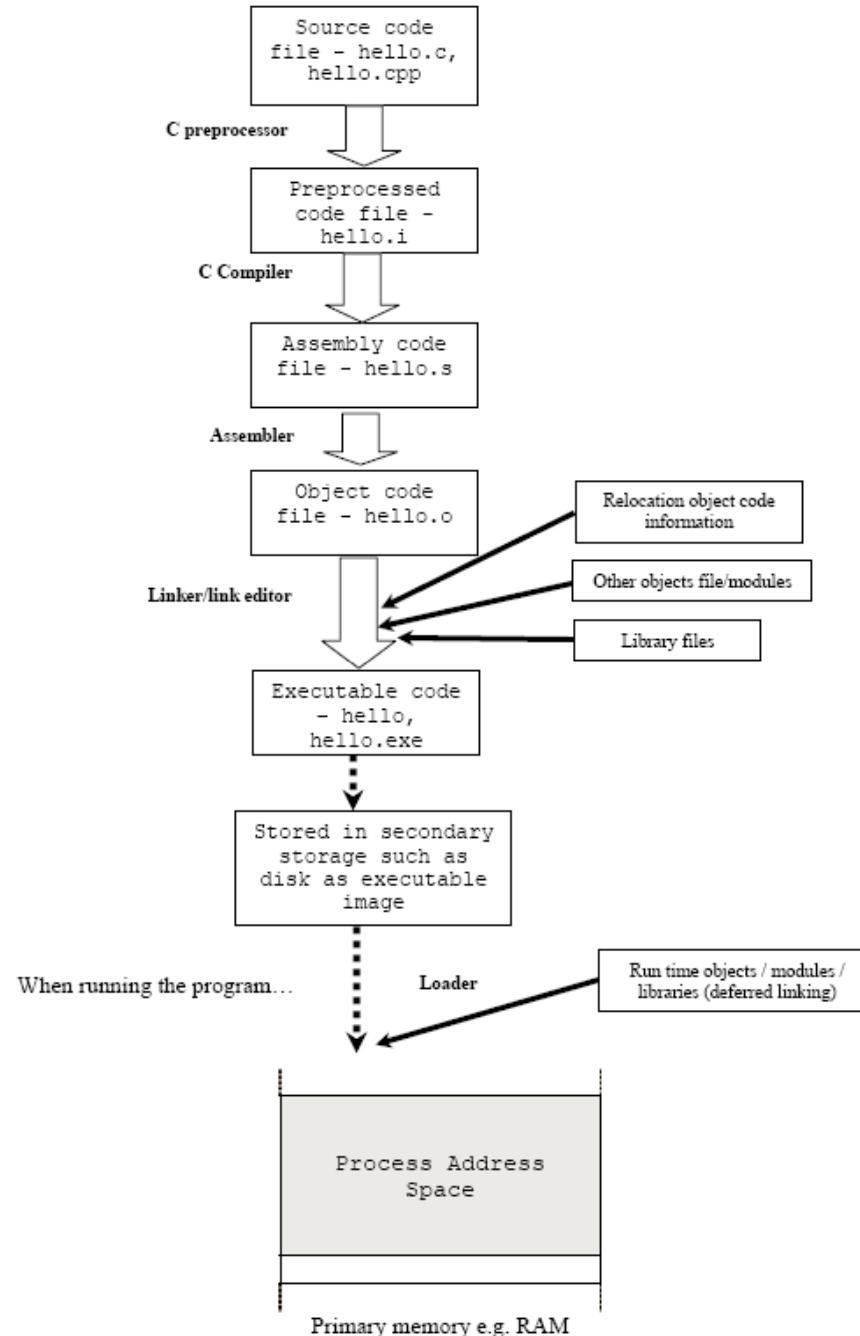
- User edits hello.c
- Preprocessing
- Compiling hello.c to hello.s
- Assembling hello.s to hello.o
- Linking hello.o + libs to hello.exe on disk
- Loader hello.exe + dll or so into RAM
- Running on CPU



# Compiler

hello.c → hello.exe

- Preprocessing
- Compiling
- Assembling
- Linking
- Loader
- Running

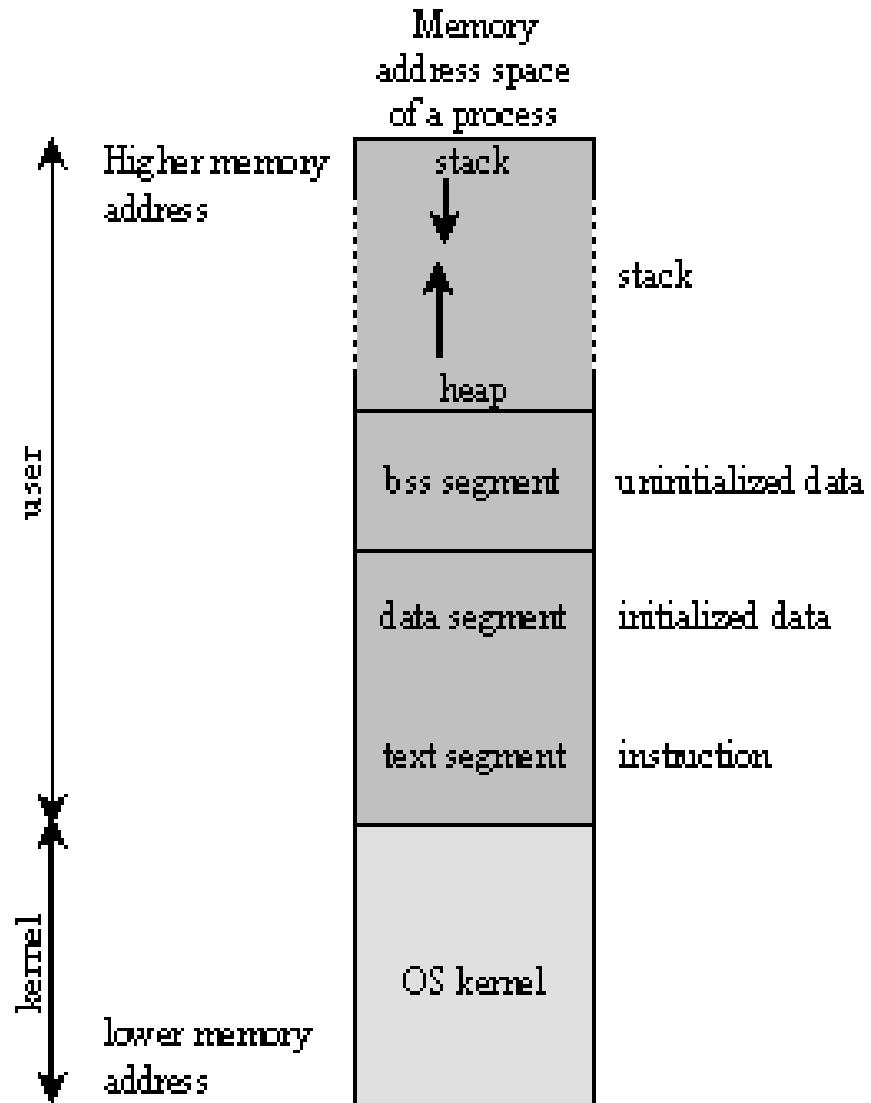


# GCC commands

- `gcc -E file.c` # run only preprocessor
- `gcc -S file.c` # creates asm file.s
- `gcc -c file.c` # compiles to file.o
- `readelf -a file.o` # (linux)
- `dumpbin /all file.o` # (windows)

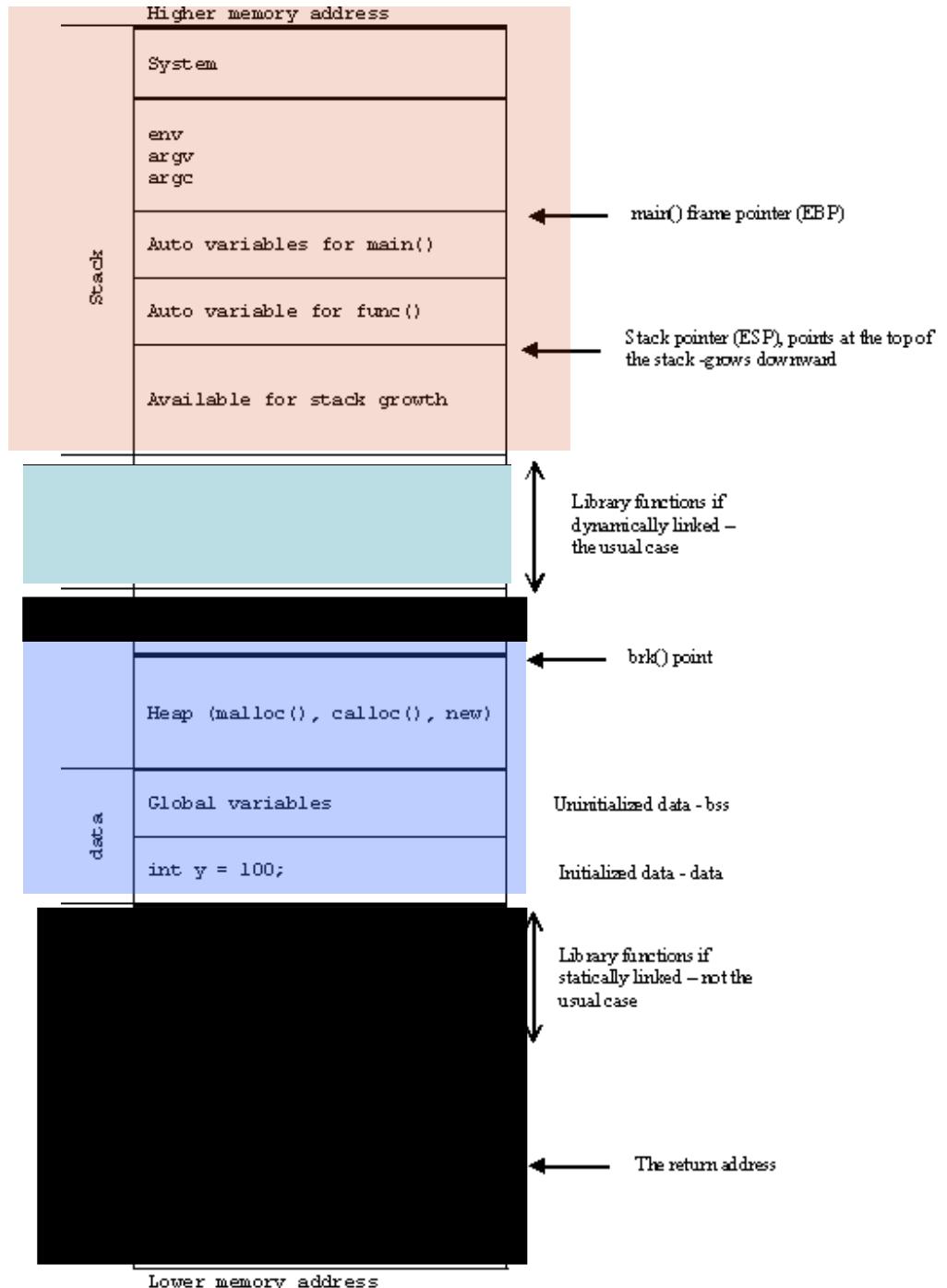
# Memory contents

- Stack (local variables, return address)
- Heap (globals, malloc)
- Data (static)
- Text (exe,dll,so)
- Kernel



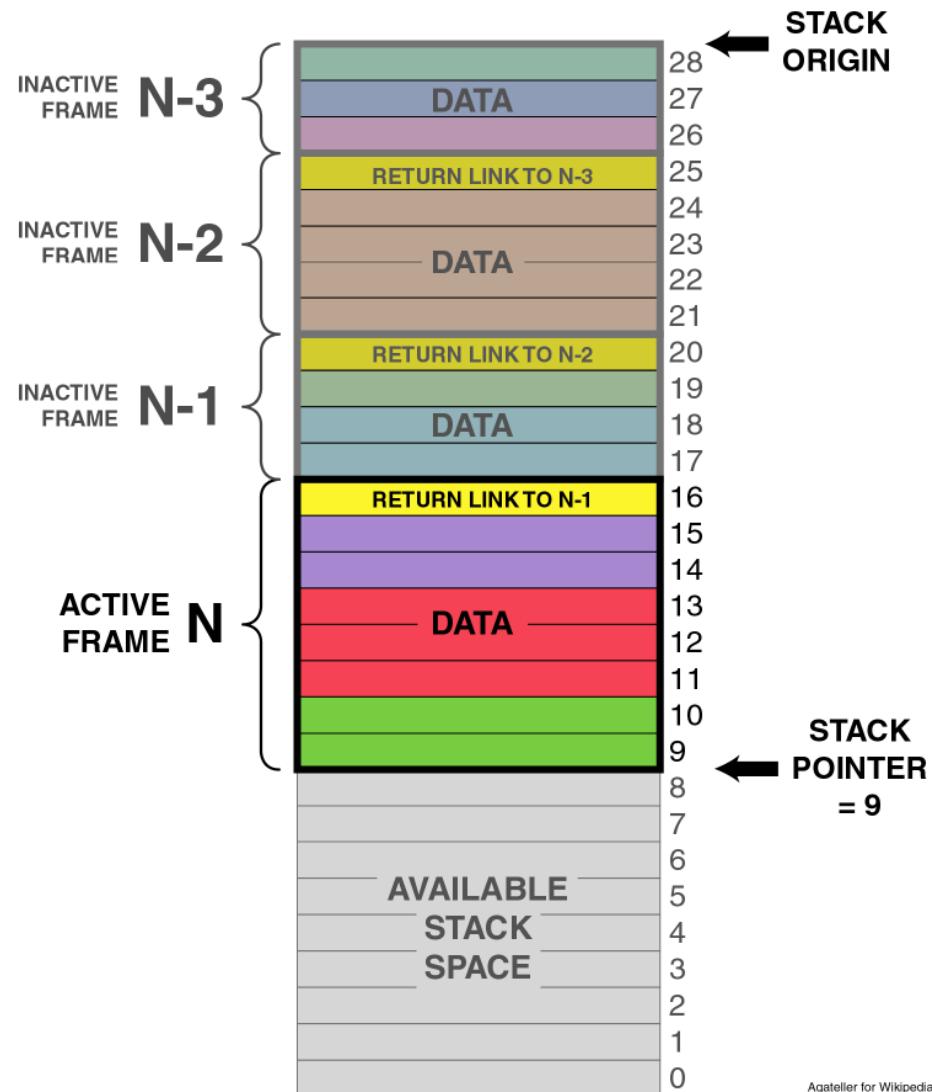
# Memory

- stack
- dll libraries
- global data
- functions



# Stack frames

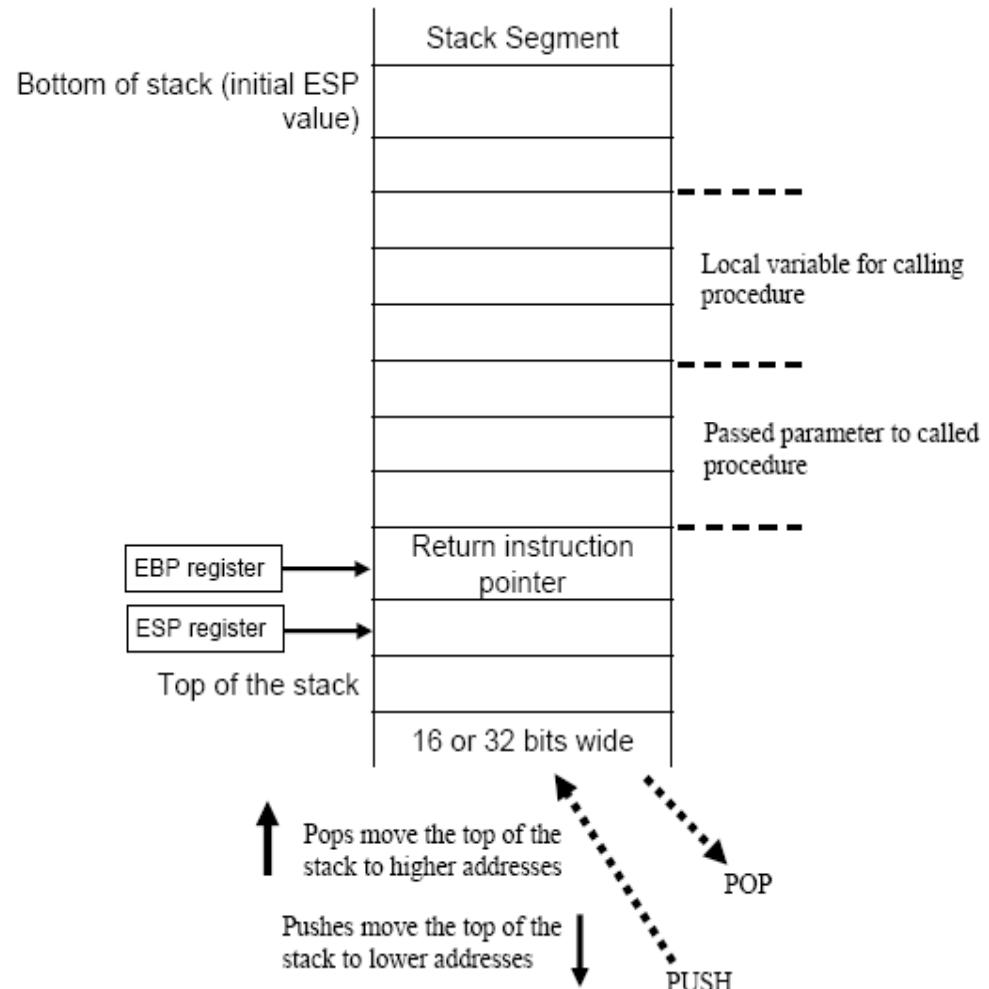
- Function **call** creates a new *frame* on the stack.
- **return** deletes the current frame, and we return to previous frame.



Agateller for Wikipedia  
Public Domain 2006

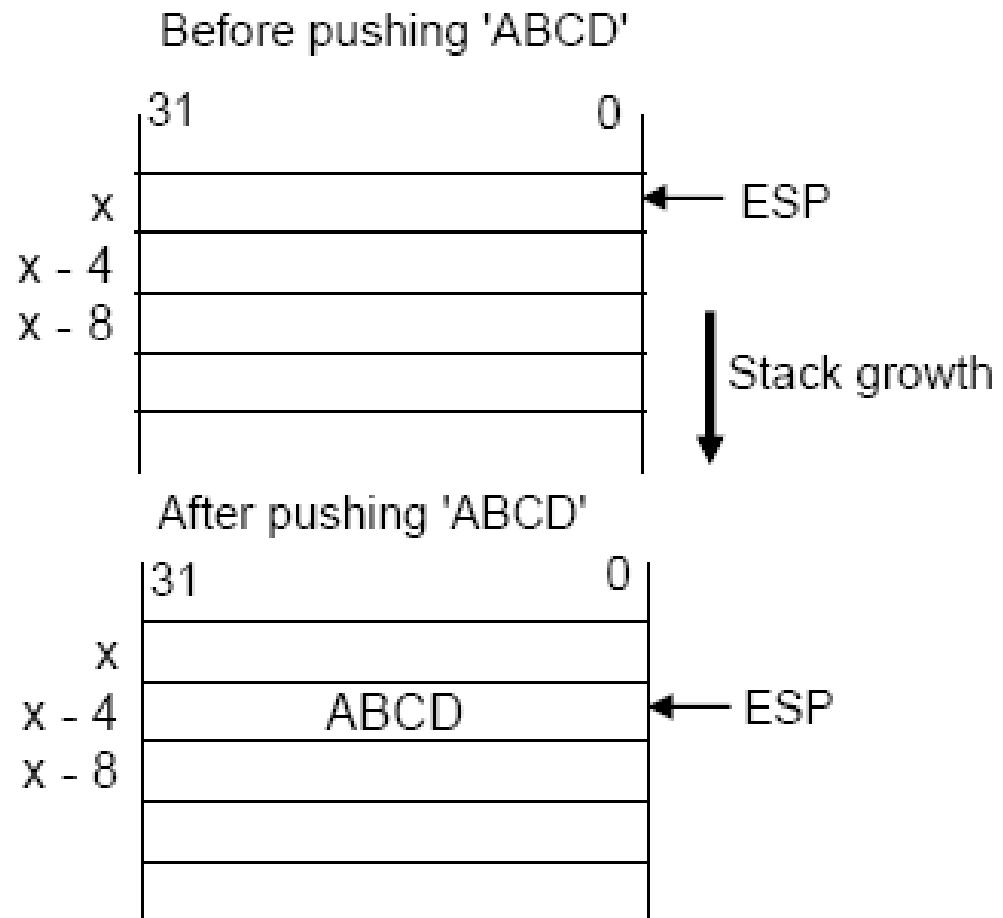
# What's in a frame on the stack?

- Local variables
- Parameters
- Return address



# push 'abcd' on the stack

- Push 'ABCD'
- Register word size is 4 bytes

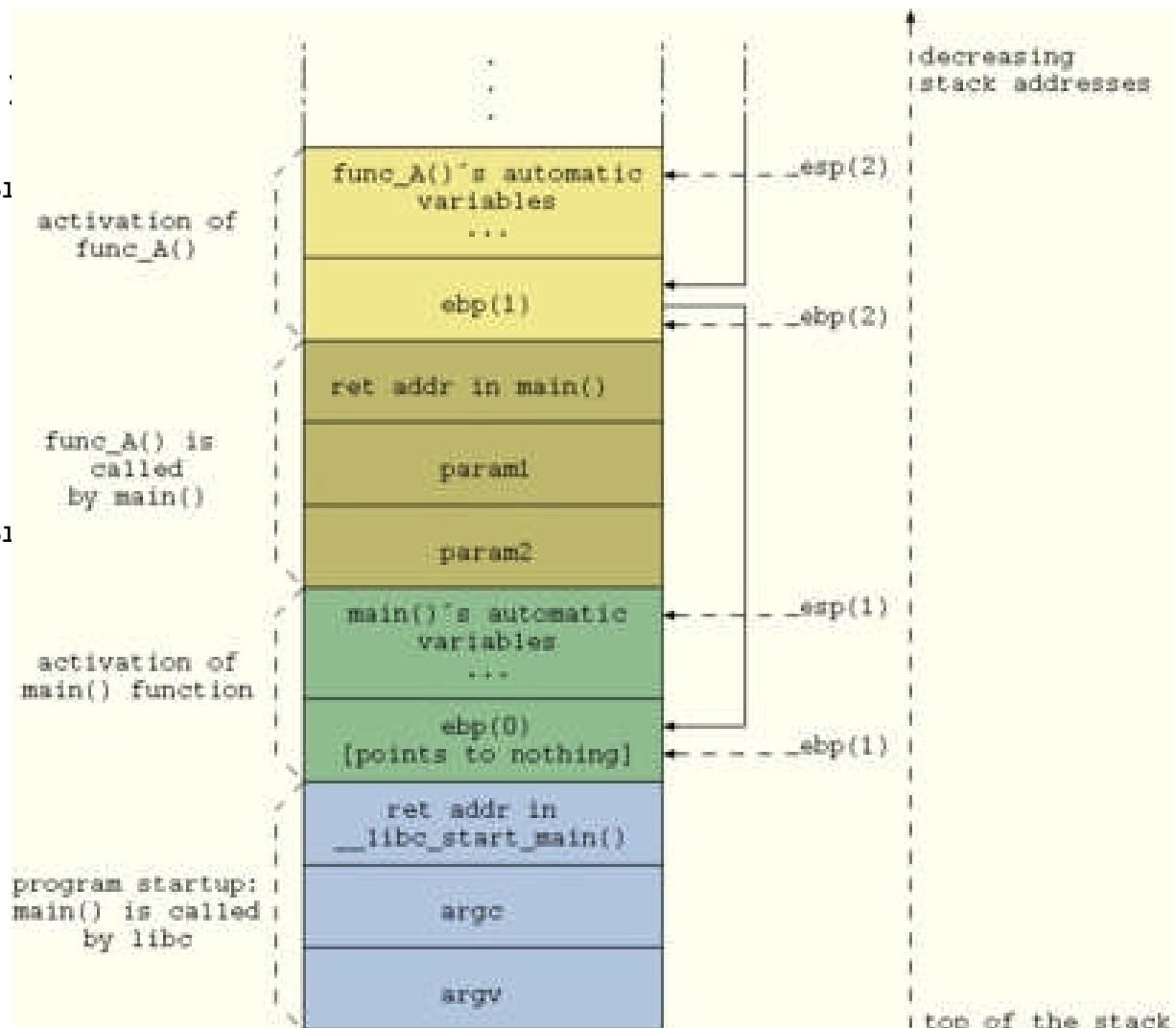


# Function call example

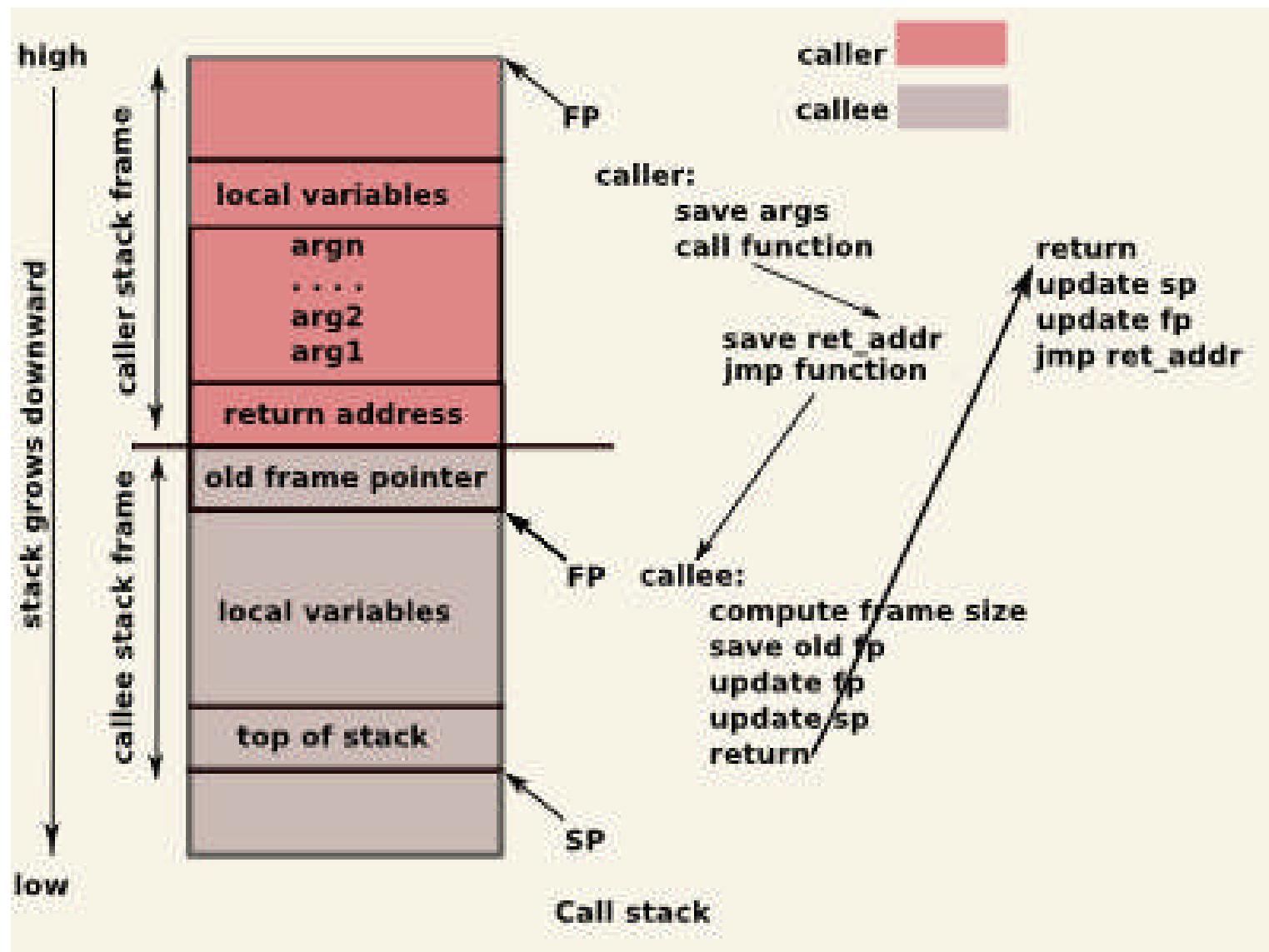
```
#include <stdio.h>

int func_A(int x, int y)
{
 int s, t; // automatic variables
 return 3;
}

int main(int argc,
 char *argv[])
{
 int i, k; // automatic variables
 func_A(1, 2);
 return 0;
}
```



# Function call



# Stack during nested function calls

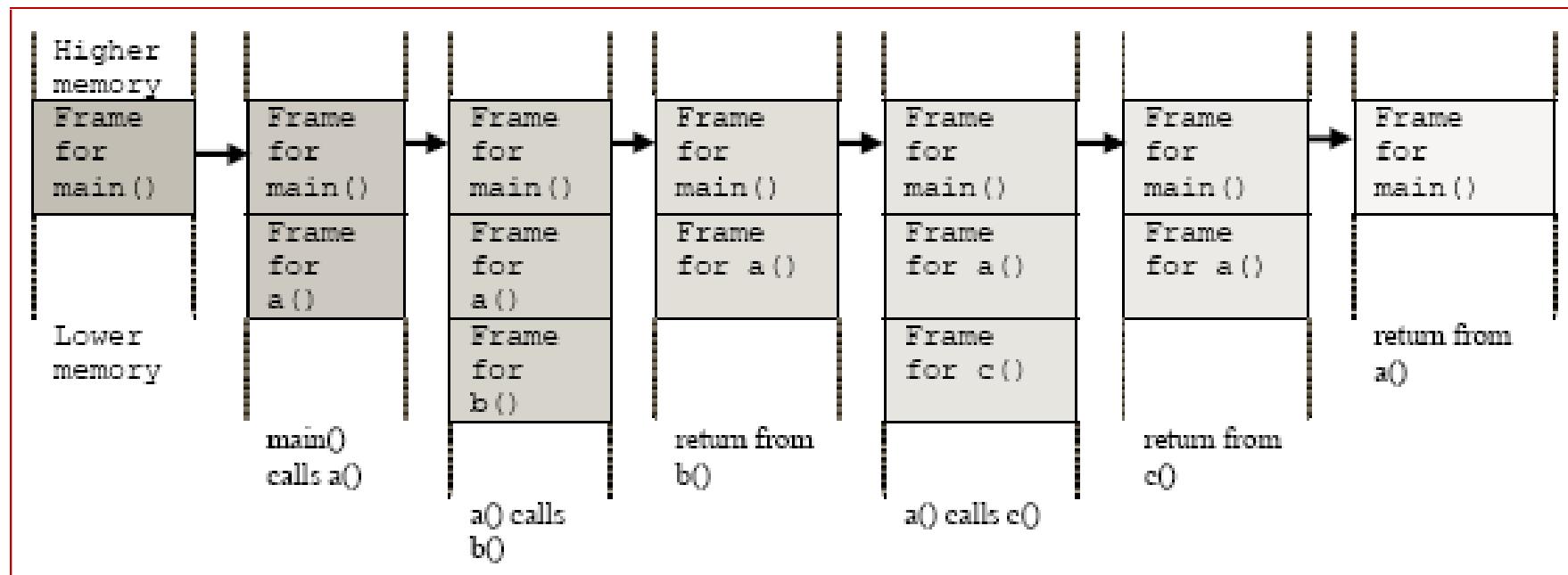
```
#include <stdio.h>

int c(){ return 'c'; }

int b(){ return 'b'; }

int a(){ b(); c(); return 'a'; }

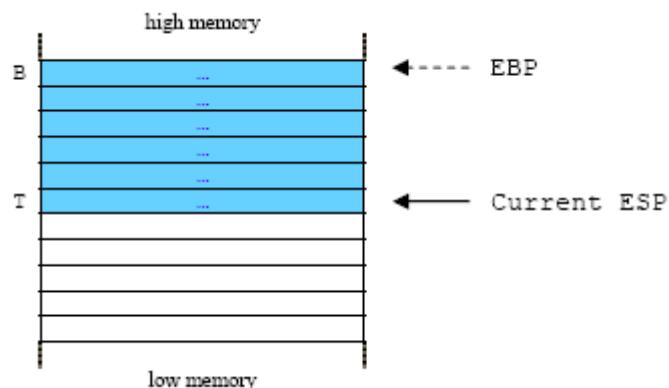
int main(){ a(); return 0; }
```



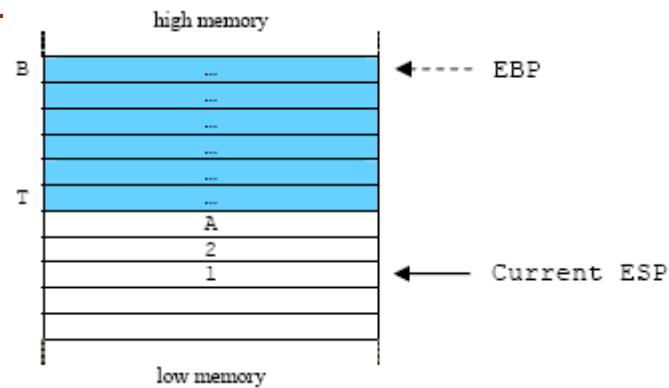
# Calling a cdecl function

- `TestFunc(1, 2, 'A'); // call`

```
0x08048388 <main+28>: movb $0x41, 0xffffffff(%ebp) ; prepare the byte of 'A'
0x0804838c <main+32>: movsbl 0xffffffff(%ebp), %eax ; put into eax
0x08048390 <main+36>: push %eax ; push the third parameter, 'A' prepared
 ; in eax onto the stack, [ebp+16]
0x08048391 <main+37>: push $0x2 ; push the second parameter, 2 onto
 ; the stack, [ebp+12]
0x08048393 <main+39>: push $0x1 ; push the first parameter, 1 onto
 ; the stack, [ebp+8]
```



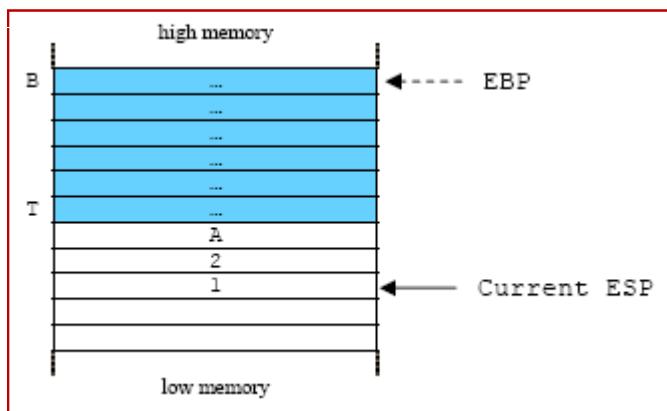
Stack before the push



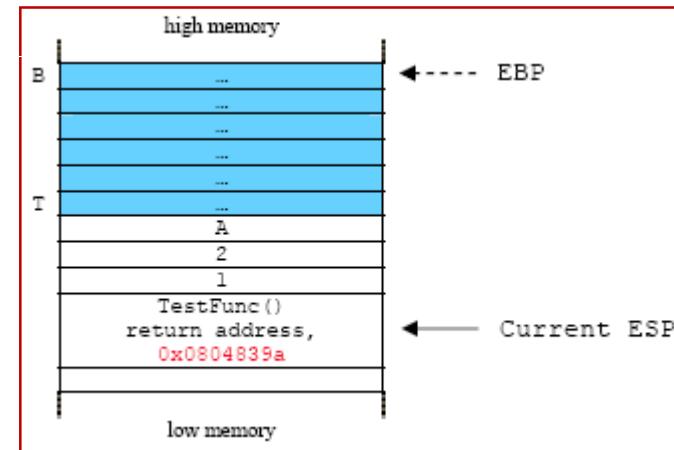
Params on the stack in reverse order

# Calling TestFunc(...)

```
0x08048395 <main+41> : ; assembler comment
 call 0x8048334 <TestFunc> ; function call.
; Push the return address [0x0804839a]
; onto the stack, [ebp+4]
```



Before

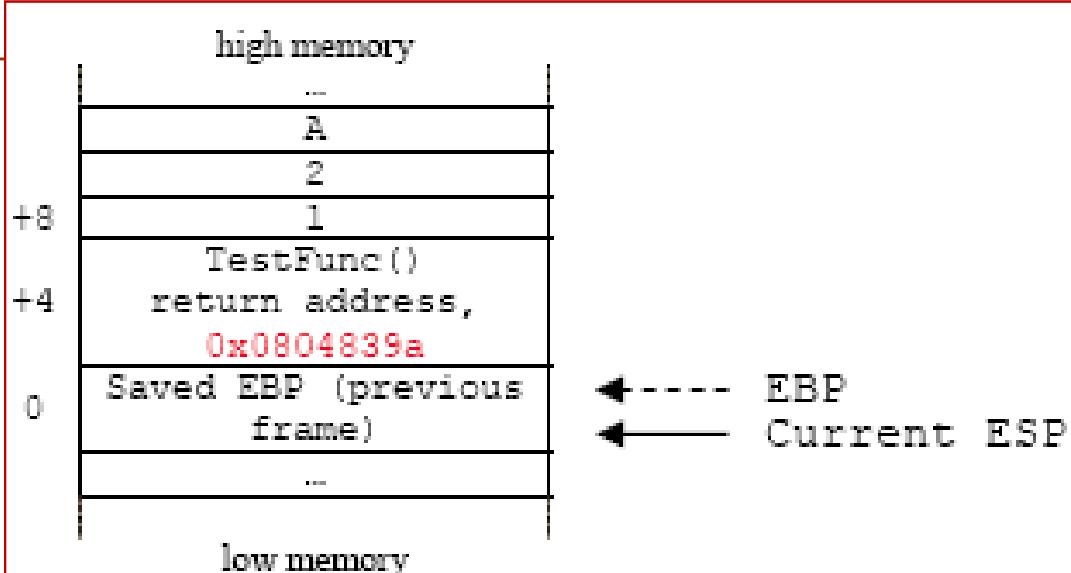


Ready to call

From <http://www.tenouk.com/Bufferoverflowc/Bufferoverflow3.html>

# Entering TestFunc(...)

```
0x08048334 <TestFunc+0>: push %ebp ;push the previous stack frame
 ;pointer onto the stack, [ebp+0]
0x08048335 <TestFunc+1>: mov %esp, %ebp ;copy the ebp into esp, now the ebp and
 esp
 ;are pointing at the same address,
 ;creating new stack frame [ebp+0]
0x08048337 <TestFunc+3>: push %edi ;save/push edi register, [ebp-4]
0x08048338 <TestFunc+4>: push %esi ;save/push esi register, [ebp-8]
0x08048339 <TestFunc+5>: sub $0x20, %esp ;subtract esp by 32 bytes for local
 ;variable and buffer if any, go to [ebp-40]
```



Pushing the EBP onto the stack, saving the previous stack frame.

# Exercises

1. Debug this file **abc.c** in CodeBlocks:

```
#include <stdio.h>

int c(){ return 'c'; }
int b(){ return 'b'; }
int a(){ b(); c(); return 'a'; }
int main(){ a(); return 0; }
```

2. Put breakpoints on all the functions
3. Run it step by step, and watch the stack.
4. Generate assembly using gcc
5. Optionally use ms-vc6 to do the same.

# Solution: gcc assembly file

```
$ gcc -S abc.c
```

```
$ cat abc.s
```

a:

```
pushl %ebp
movl %esp, %ebp
call _b
call _c
movl $97, %eax ; return 'a'
popl %ebp
ret
```

. main:

```
pushl %ebp
movl %esp, %ebp
andl $-16, %esp
call __main
call _a
movl $0, %eax
movl %ebp, %esp
popl %ebp
ret
```

# Solution: vc6 assembly file

```
c:\tmp> cl /FAs abc.c
c:\tmp> cat abc.asm
; line 4 : int a(){ b(); c(); return 'a'; }
 push ebp
 mov ebp, esp
 call _b
 call _c
 mov eax, 97; 00000061H
 pop ebp
 ret 0
```

---

```
c:\tmp> cl /Fc abc.c
c:\tmp> cat abc.cod
; line 4 : int a(){ b(); c(); return 'a'; }
00014 55 push ebp
00015 8b ec mov ebp, esp
00017 e8 00 00 00 00 call _b
0001c e8 00 00 00 00 call _c
00021 b8 61 00 00 00 mov eax, 97; 00000061H
00026 5d pop ebp
00027 c3 ret 0
```

# Stack of abc.c in Codeblocks

The screenshot shows the Codeblocks IDE interface with the following windows:

- main.c**: The code editor window containing the following C code:

```
#include <stdio.h>
int c() { return 'c'; }
int b() { return 'b'; }
int a() { b(); c(); return 'a'; }
int main() { a(); return 0; }
```
- Call stack**: A table showing the call stack with three entries:

| Nr | Address  | Fun... | File    |
|----|----------|--------|---------|
| 0  |          | b()    | C:\temp |
| 1  | 00401350 | a()    | C:\temp |
| 2  | 0040136C | main() | C:\temp |
- CPU Registers**: A table showing CPU register values:

| Re... |          |
|-------|----------|
| eax   |          |
| ecx   |          |
| edx   | 0x77c61  |
| ebx   | 0x7ffd1  |
| esp   | 0022FF48 |
- Disassembly**: The assembly output for function b, starting at frame 0022FF48:

```
Function: b (C:\temp\x\main.c:3)
Frame start: 0022FF48
0x0040133E push %ebp
0x0040133F mov %esp,%ebp
0x00401341 mov $0x62,%eax
0x00401346 pop %ebp
0x00401347 ret
```

# MAPLE

## mathematical software



Maple V Release  
5.1

# Maple

## Symbolic and Numerical Mathematics

C:\> start C:\tools\MAPLEV4\BIN.WIN\WMAPLE32.exe

Maple V Release 4 - [Untitled (1)]

File Edit View Insert Format Options Window Help

[> evalf(sin(Pi/4));  
>  
0.7071067810  
> factor(6\*x^2+18\*x-24);  
6 (x + 4) (x - 1)  
> expand((x+1)/(x+2));  
$$\frac{x}{x + 2} + \frac{1}{x + 2}$$
  
> diff(ln(x^4-22),x);  
$$4 \frac{x^3}{x^4 - 22}$$
  
> 100!;  
>  
9332621544394415268169923885626670049071596826438162146859296389521  
1759999322991560894146397615651828625369792082722375825118521091681  
6400000000000000000000000000000000C000  
>

# CRT: Chinese Remainder Theorem

Solve for  $x$ , given these modular equations:

- $x \equiv 2 \pmod{3}$
- $x \equiv 3 \pmod{5}$
- $x \equiv 2 \pmod{7}$

```
C:\> c:\tools\maplev4\bin.win\cmaple.exe
>> ?? # General help
>> ??chinese # Search for help on CRT
Calling Sequence: chrem(u, m)
Parameters: u - the list of evaluations [u0, u1,..., un]
 m - the list of moduli [m0,m1,...,mn]
> chrem([2,3,2], [3,5,7]);
23
> quit;
C:\>
```

# Powermod (used in RSA)

# Ordinary arithmetic power and then mod

>  $7^{**}123456 \bmod 101;$

16

# Powermod is faster and handles larger input

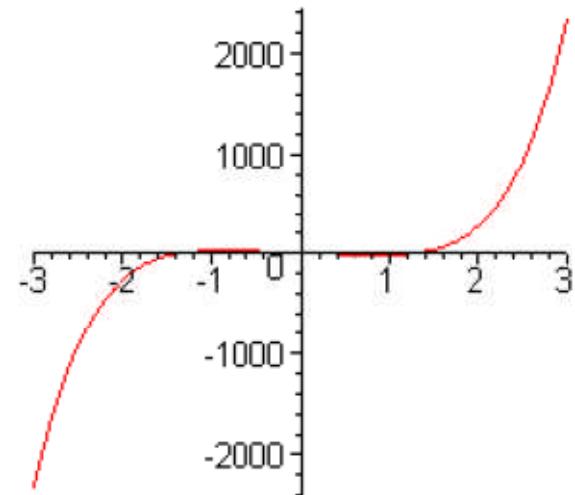
>  $7 \&^ 1234567891 \bmod 101;$

98

# Graphing

```
> f := x -> 10*x^5 - 30*x +10;
> plot(f,-3..3);
```

- . Stmt end in semi-colon
- . Assignment is "colon equal"



# Numerical Computation

```
> Pi^3^2
```

9 Pi # Maple keeps it as symbols.

```
> evalf(Pi^3^2,30);
```

28.2743338823081391461637904495

```
> (3+2*I)*(2-I)/(3-I); # Complex Numbers
```

$$\frac{23}{10} + \frac{11}{10}I$$

$$\begin{bmatrix} > (3+2*I)*(2-I)/(3-I) ; \\ \frac{23}{10} + \frac{11}{10}I \end{bmatrix}$$

# Quadratic Equations

```
> solve(3*x^2+2*x +1,x);
```

```
-1/3+1/3*I*2^(1/2), -1/3-1/3*I*2^(1/2)
```

```
[> solve(3*x^2+2*x +1 ,x) ;
 - 1/3+1/3 I √2, - 1/3-1/3 I √2]
```

# LaTeX

## Typesetting Software (not a wordprocessor)

LATEX



# LaTeX

- **LaTeX** is a document preparation system for high-quality typesetting, free software written by Leslie Lamport, built on top of Tex and metafont by Knuth.
- It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing.
- **LaTeX** is *not* a word processor!
- **LaTeX** encourages authors *not* to worry about the appearance of their documents but to concentrate on getting the right content.
- This has changed as MS Word has improved, especially for including multimedia in documents.

# Installation

- Already installed on Linux
- Windows: download and install **MikTex** in **C : \miktex**

For LaTeX commands:

- *search google "latex tex tutorial"*

*Or start here:*

- *<http://www.tug.org/begin.html>*

# Sample Document

c:\> cat test.tex

```
\documentclass{article}
\title{Algorithm assignment}
\author{Mosh Ahmed}
\date{29 February 2013}
\begin{document}
 \maketitle
 Hello world!
\end{document}
```

# Compiling file.tex to file.dvi

C:\> latex test.tex

...

Output written on test.dvi (1 page, 424 bytes).

Transcript written on test.log.

# View the dvi file

C:\> yap test.dvi .. View the output

# Printing dvi file

```
C:\> dvipdfm test.dvi # dvi to pdf
test.dvi -> test.pdf
```

# Greek letters and Symbols

- Search google for Latex manual in pdf.

|               |                          |             |                        |             |                        |            |                       |
|---------------|--------------------------|-------------|------------------------|-------------|------------------------|------------|-----------------------|
| $\alpha$      | <code>\alpha</code>      | $\theta$    | <code>\theta</code>    | $\circ$     | <code>\circ</code>     | $\tau$     | <code>\tau</code>     |
| $\beta$       | <code>\beta</code>       | $\vartheta$ | <code>\vartheta</code> | $\pi$       | <code>\pi</code>       | $\upsilon$ | <code>\upsilon</code> |
| $\gamma$      | <code>\gamma</code>      | $\iota$     | <code>\iota</code>     | $\varpi$    | <code>\varpi</code>    | $\phi$     | <code>\phi</code>     |
| $\delta$      | <code>\delta</code>      | $\kappa$    | <code>\kappa</code>    | $\rho$      | <code>\rho</code>      | $\varphi$  | <code>\varphi</code>  |
| $\epsilon$    | <code>\epsilon</code>    | $\lambda$   | <code>\lambda</code>   | $\varrho$   | <code>\varrho</code>   | $\chi$     | <code>\chi</code>     |
| $\varepsilon$ | <code>\varepsilon</code> | $\mu$       | <code>\mu</code>       | $\sigma$    | <code>\sigma</code>    | $\psi$     | <code>\psi</code>     |
| $\zeta$       | <code>\zeta</code>       | $\nu$       | <code>\nu</code>       | $\varsigma$ | <code>\varsigma</code> | $\omega$   | <code>\omega</code>   |
| $\eta$        | <code>\eta</code>        | $\xi$       | <code>\xi</code>       |             |                        |            |                       |
| $\Gamma$      | <code>\Gamma</code>      | $\Lambda$   | <code>\Lambda</code>   | $\Sigma$    | <code>\Sigma</code>    | $\Psi$     | <code>\Psi</code>     |
| $\Delta$      | <code>\Delta</code>      | $\Xi$       | <code>\Xi</code>       | $\Upsilon$  | <code>\Upsilon</code>  | $\Omega$   | <code>\Omega</code>   |
| $\Theta$      | <code>\Theta</code>      | $\Pi$       | <code>\Pi</code>       | $\Phi$      | <code>\Phi</code>      |            |                       |

# Arrows

$\geq \ \backslash geq \ \gg \ \backslash gg \ \leq \ \backslash leq \ \ll \ \backslash ll \ \neq \ \backslash neq$

|                       |                                  |                        |                                   |                    |                               |
|-----------------------|----------------------------------|------------------------|-----------------------------------|--------------------|-------------------------------|
| $\Downarrow$          | <code>\Downarrow</code>          | $\Longleftarrow$       | <code>\longleftarrow</code>       | $\nwarrow$         | <code>\nwarrow</code>         |
| $\downarrow$          | <code>\downarrow</code>          | $\Leftarrow$           | <code>\Leftarrow</code>           | $\Rightarrow$      | <code>\Rightarrow</code>      |
| $\leftarrowtail$      | <code>\leftarrowtail</code>      | $\Longleftarrowtail$   | <code>\Longleftarrowtail</code>   | $\rightarrowtail$  | <code>\rightarrowtail</code>  |
| $\hookleftarrow$      | <code>\hookleftarrow</code>      | $\Longleftarrowtail$   | <code>\Longleftarrowtail</code>   | $\rightarrowtail$  | <code>\rightarrowtail</code>  |
| $\hookrightarrow$     | <code>\hookrightarrow</code>     | $\Longrightarrowtail$  | <code>\Longrightarrowtail</code>  | $\searrowtail$     | <code>\searrowtail</code>     |
| $\leadsto^*$          | <code>\leadsto^*</code>          | $\longmapsto$          | <code>\longmapsto</code>          | $\swarrowtail$     | <code>\swarrowtail</code>     |
| $\leftarrowtail$      | <code>\leftarrowtail</code>      | $\Longrightarrowtail$  | <code>\Longrightarrowtail</code>  | $\uparrowtail$     | <code>\uparrowtail</code>     |
| $\Leftarrowtail$      | <code>\Leftarrowtail</code>      | $\Longrightarrowtail$  | <code>\Longrightarrowtail</code>  | $\uparrowtail$     | <code>\uparrowtail</code>     |
| $\Leftrightarrowtail$ | <code>\Leftrightarrowtail</code> | $\mapsto$              | <code>\mapsto</code>              | $\updownarrowtail$ | <code>\updownarrowtail</code> |
| $\leftrightharpoonup$ | <code>\leftrightharpoonup</code> | $\nearrowtail^\dagger$ | <code>\nearrowtail^\dagger</code> | $\Updownarrowtail$ | <code>\Updownarrowtail</code> |

`\arccos \cos \csc \exp \ker \limsup \min \sinh  
 \arcsin \cosh \deg \gcd \lg \ln \Pr \sup  
 \arctan \cot \det \hom \lim \log \sec \tan  
 \arg \coth \dim \inf \liminf \max \sin \tanh`

$\pi \ \backslash pi \ \rho \ \backslash rho$   
 $\varpi \ \backslash varpi \ \varrho \ \backslash varrho$   
 $\varvarpi \ \backslash varvarpi \ \varvarrho \ \backslash varvarrho$

# Vectors

|                       |                                   |                          |                                    |
|-----------------------|-----------------------------------|--------------------------|------------------------------------|
| $\widetilde{abc}$     | <code>\widetilde{abc}*{}</code>   | $\widehat{abc}$          | <code>\widehat{abc}*{}</code>      |
| $\overleftarrow{abc}$ | <code>\overleftarrow{abc}†</code> | $\overrightarrow{abc}$   | <code>\overrightarrow{abc}†</code> |
| $\overline{abc}$      | <code>\overline{abc}</code>       | $\underline{abc}$        | <code>\underline{abc}</code>       |
| $\overbrace{abc}$     | <code>\overbrace{abc}</code>      | $\underbrace{abc}$       | <code>\underbrace{abc}</code>      |
| $\sqrt{abc}$          |                                   | <code>\sqrt{abc}‡</code> |                                    |

|              |                         |              |                           |            |                       |              |                         |
|--------------|-------------------------|--------------|---------------------------|------------|-----------------------|--------------|-------------------------|
| $\aleph$     | <code>\aleph</code>     | $\diamond$   | <code>\Diamond*</code>    | $\infty$   | <code>\infty</code>   | $'$          | <code>\prime</code>     |
| $\angle$     | <code>\angle</code>     | $\lozenge$   | <code>\diamondsuit</code> | $\mho$     | <code>\mho*</code>    | $\sharp$     | <code>\sharp</code>     |
| $\backslash$ | <code>\backslash</code> | $\emptyset$  | <code>\emptyset</code>    | $\nabla$   | <code>\nabla</code>   | $\spadesuit$ | <code>\spadesuit</code> |
| $\Box$       | <code>\Box*†</code>     | $\flat$      | <code>\flat</code>        | $\natural$ | <code>\natural</code> | $\surd$      | <code>\surd</code>      |
| $\clubsuit$  | <code>\clubsuit</code>  | $\heartsuit$ | <code>\heartsuit</code>   | $\neg$     | <code>\neg</code>     | $\triangle$  | <code>\triangle</code>  |

# Math

TABLE 67: Binary Relations

|           |                      |             |                        |           |                      |           |                      |
|-----------|----------------------|-------------|------------------------|-----------|----------------------|-----------|----------------------|
| $\approx$ | <code>\approx</code> | $\equiv$    | <code>\equiv</code>    | $\perp$   | <code>\perp</code>   | $\smile$  | <code>\smile</code>  |
| $\asymp$  | <code>\asymp</code>  | $\sim$      | <code>\frown</code>    | $\prec$   | <code>\prec</code>   | $\succ$   | <code>\succ</code>   |
| $\bowtie$ | <code>\bowtie</code> | $\bowtie$   | <code>\Join^*</code>   | $\preceq$ | <code>\preceq</code> | $\succeq$ | <code>\succeq</code> |
| $\cong$   | <code>\cong</code>   | $\mid$      | $\mid$                 | $\propto$ | <code>\propto</code> | $\vdash$  | <code>\vdash</code>  |
| $\dashv$  | <code>\dashv</code>  | $\models$   | <code>\models</code>   | $\sim$    | <code>\sim</code>    |           |                      |
| $\doteq$  | <code>\doteq</code>  | $\parallel$ | <code>\parallel</code> | $\simeq$  | <code>\simeq</code>  |           |                      |

# Latex Math Examples

% Example 1

```
\begin{equation*}R = \frac{\displaystyle\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2 \right]^{1/2}}\end{equation*}
```

% Example 2

```
$$c = \sqrt{a^2 + b^2} \\ \int_{-\infty}^{\infty} \frac{1}{x} dx \\ f(x) = \sum_{n=0}^{\infty} \alpha_n x^n \\ x_{1,2} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a} \\ \hat{a} \bar{b} \vec{c} \mathbf{x}' \dot{\mathbf{x}} \ddot{\mathbf{x}} \\ $$
```

$$R = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\left[ \sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2 \right]^{1/2}}$$

$$c = \sqrt{a^2 + b^2}$$

$$\int_{-\infty}^{\infty} \frac{1}{x} dx$$

$$f(x) = \sum_{n=0}^{\infty} \alpha_n x^n$$

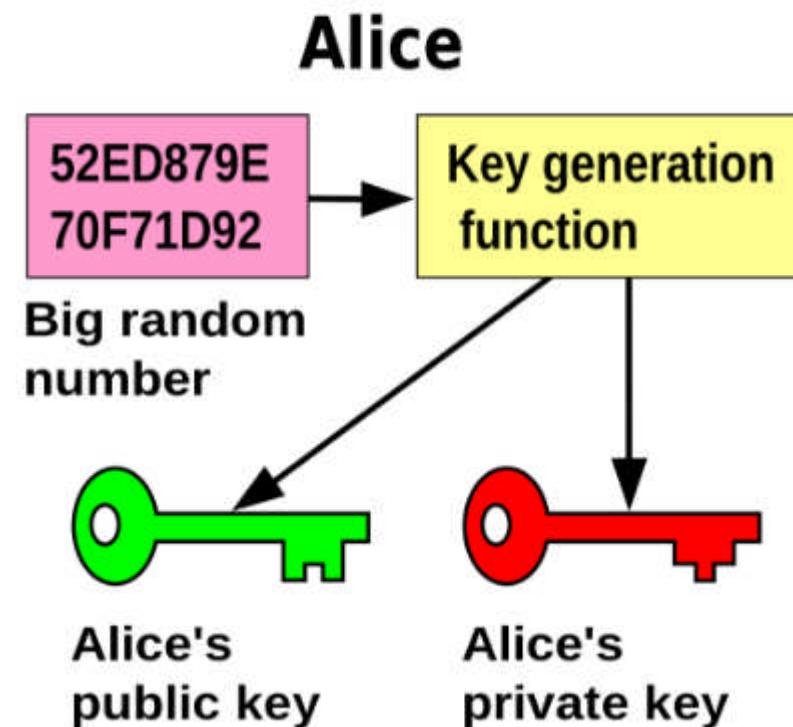
$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\hat{a} \bar{b} \vec{c} \mathbf{x}' \dot{\mathbf{x}} \ddot{\mathbf{x}}$$

# Using SSH

1. Generating your keypair
2. Distributing public key
3. Using private key for ssh login

# Key Gen

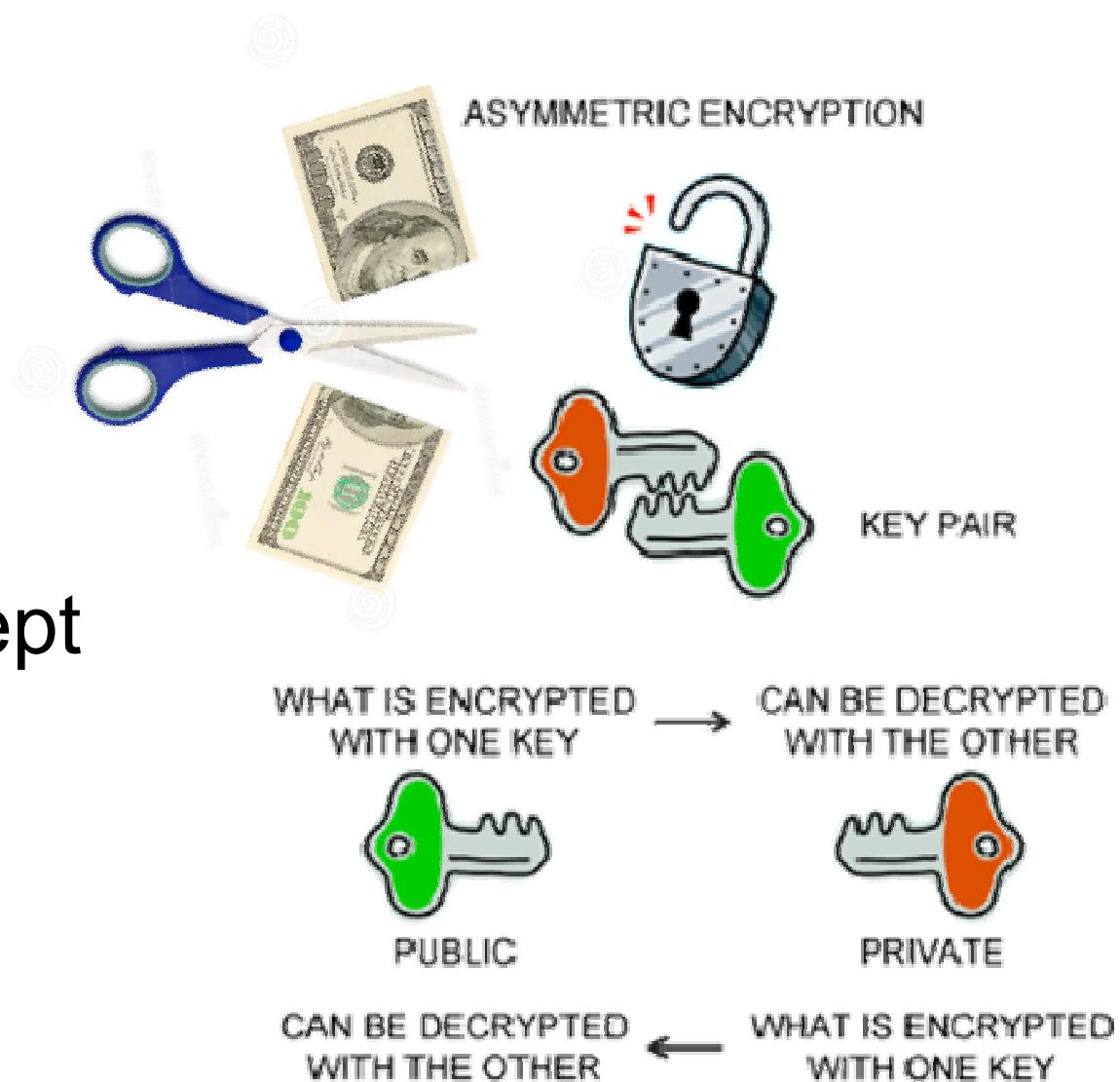


# Your keypair(s)

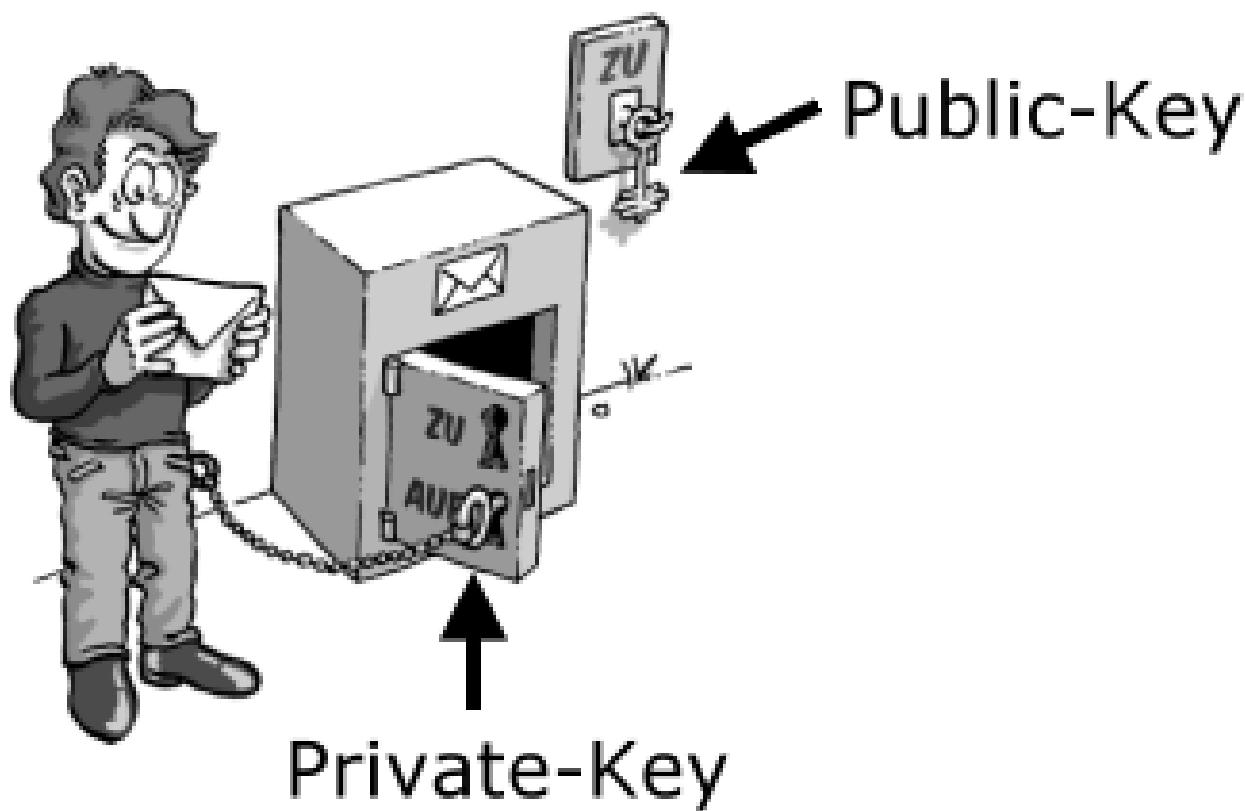
## Keypair

- publickey
- privatekey

Privatekey is kept  
locked with  
passphrase



# Key pair



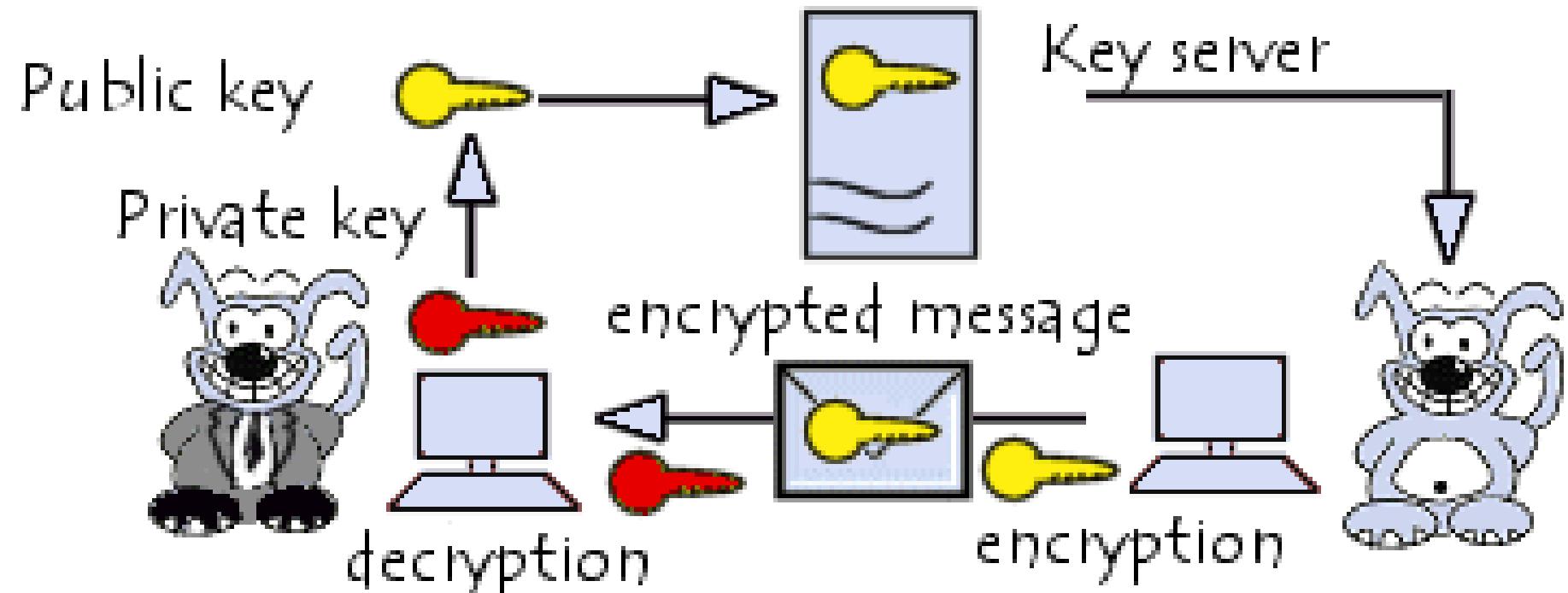
# Passphrase protect your keypair

Private Keys

Pass Phrase



# Using keypair



# Two way to use KeyPair

## Asymmetric Cryptography

*One key encrypts the data, and the other decrypts it.*

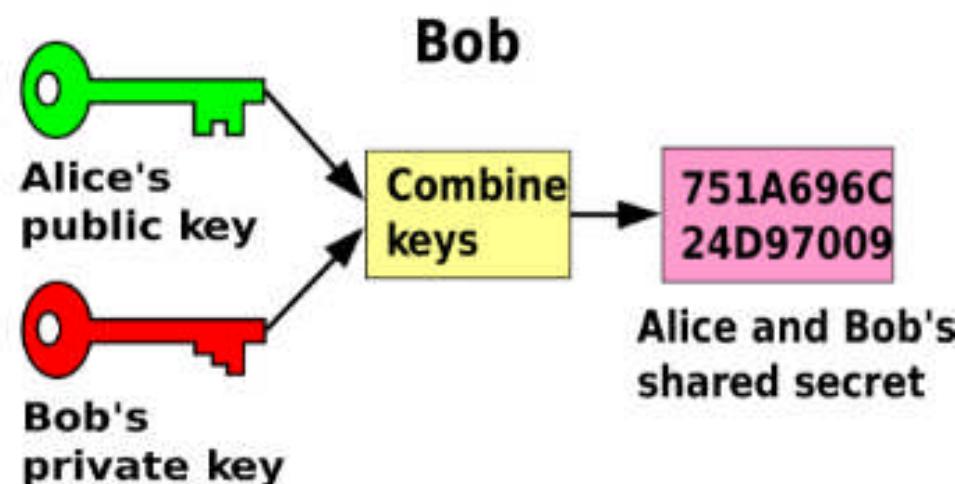
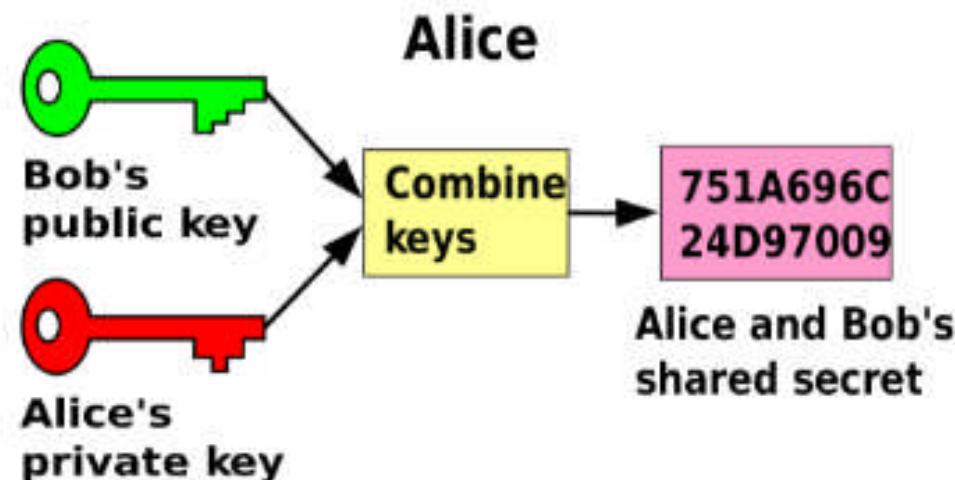


Confidentiality Mode

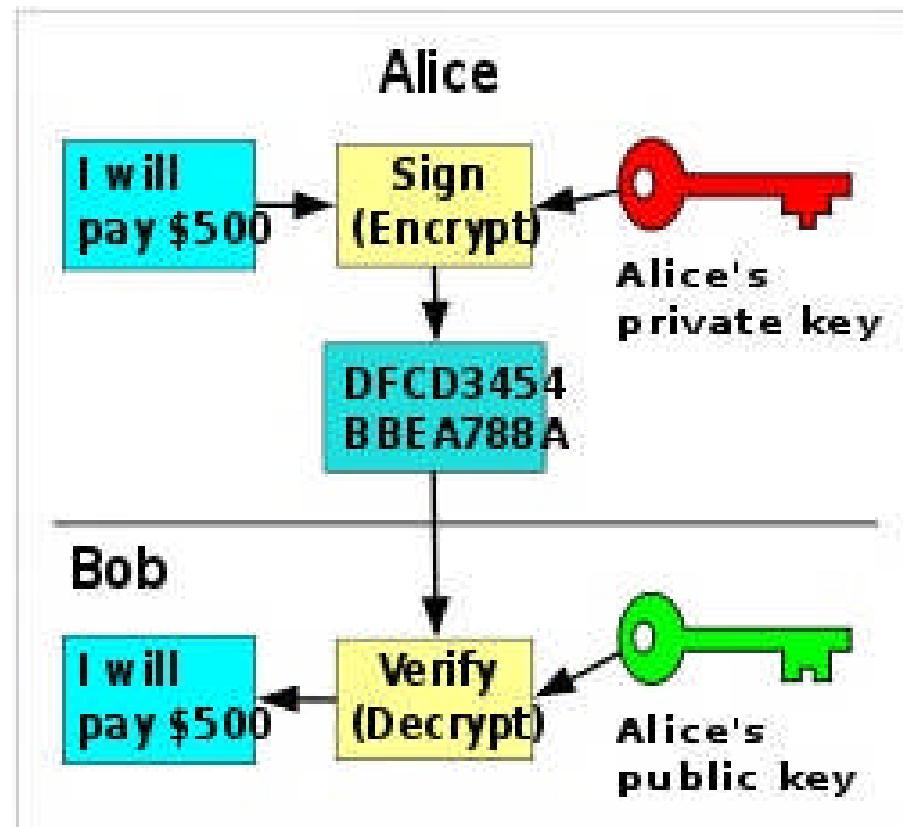
Authentication Mode



# Shared Secret



# Signing



In some related signature schemes, the private key is used to sign a message; anyone can check the signature using the public key. Validity depends on security of the private key.

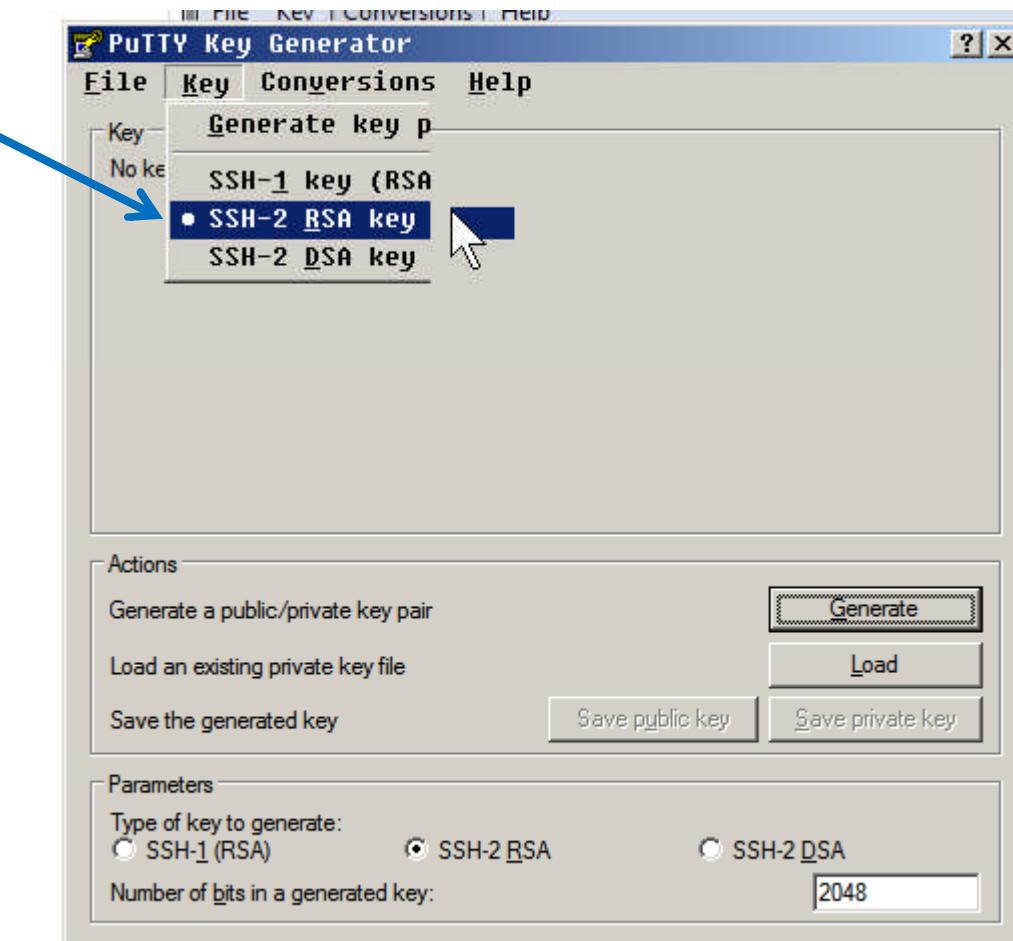
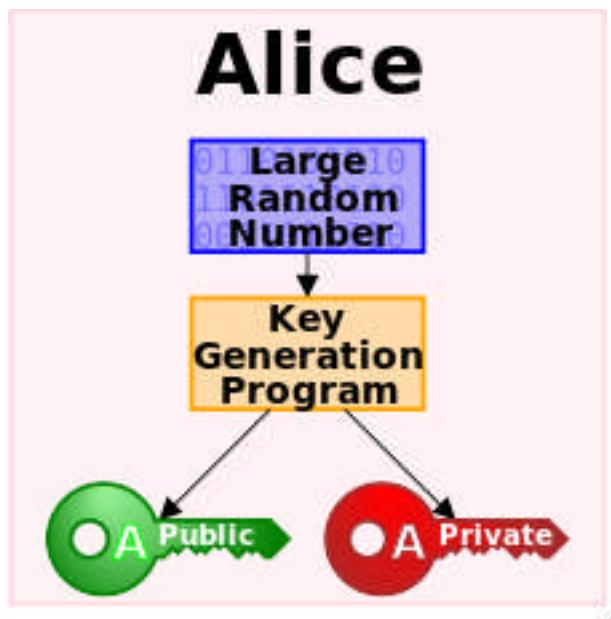
# Tools using Keypairs

- ssl (https)
- ssh, putty
- filezilla
- scp, winscp
- rsync
- pgp, gpg



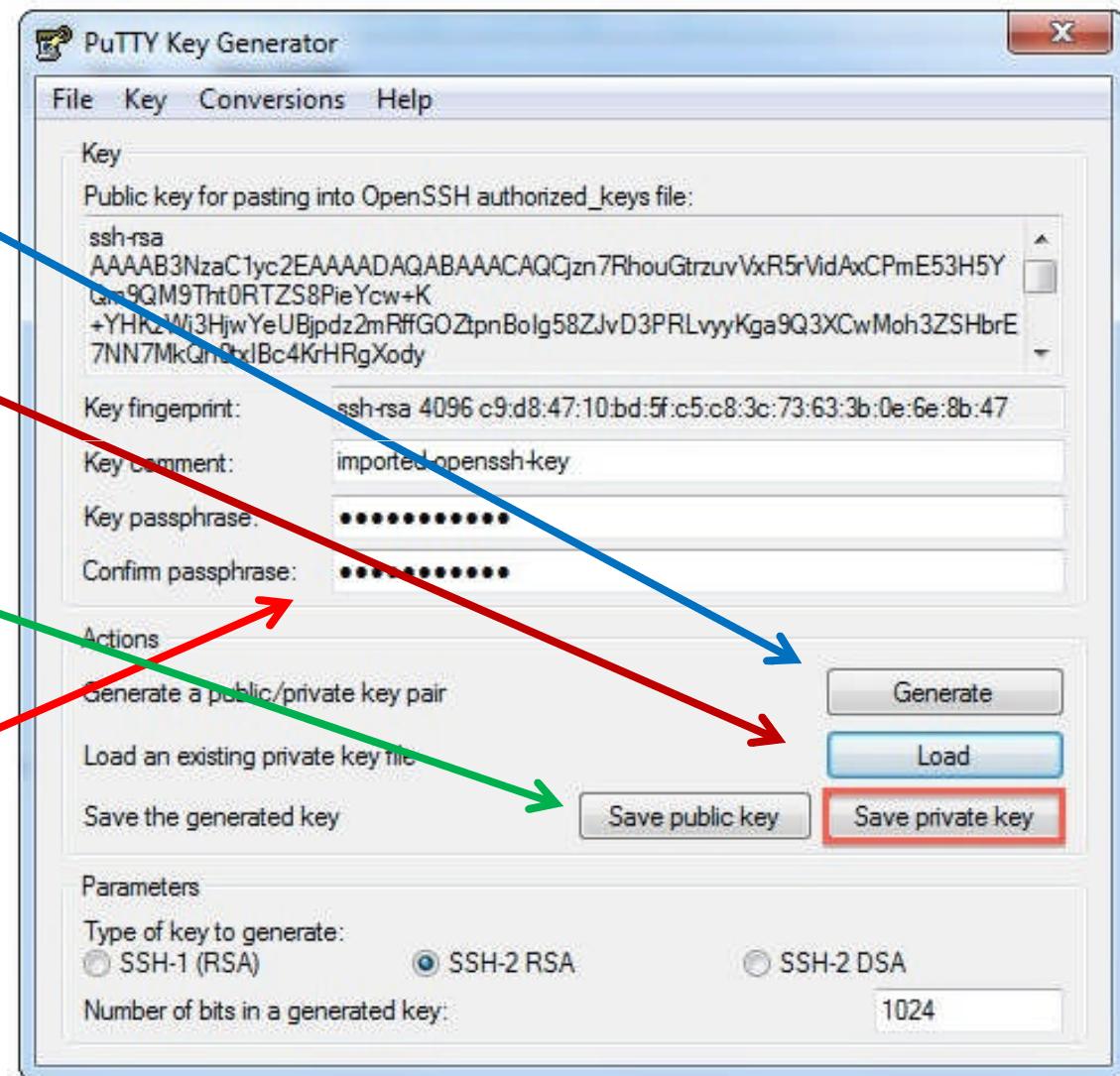
# Generating keypair with PuttyGen on Windows

## 1. puttygen



# Generating keypair with PuttyGen on Windows

1. puttygen
2. save  
**prvkey.ppk**
3. save  
**pubkey.txt**
4. Remember  
**passphrase**



# Save ppk as openssh keys also



# Create ssh keys on Linux

```
~/.ssh> ssh-keygen -t rsa -C KEYNAME
```

```
~/.ssh> ls
```

id\_rsa .. ssh keypair

(prvkey in kfile is locked with passphrase).

id\_rsa.pub .. ssh pubkey

putty-pub.txt .. putty pubkey

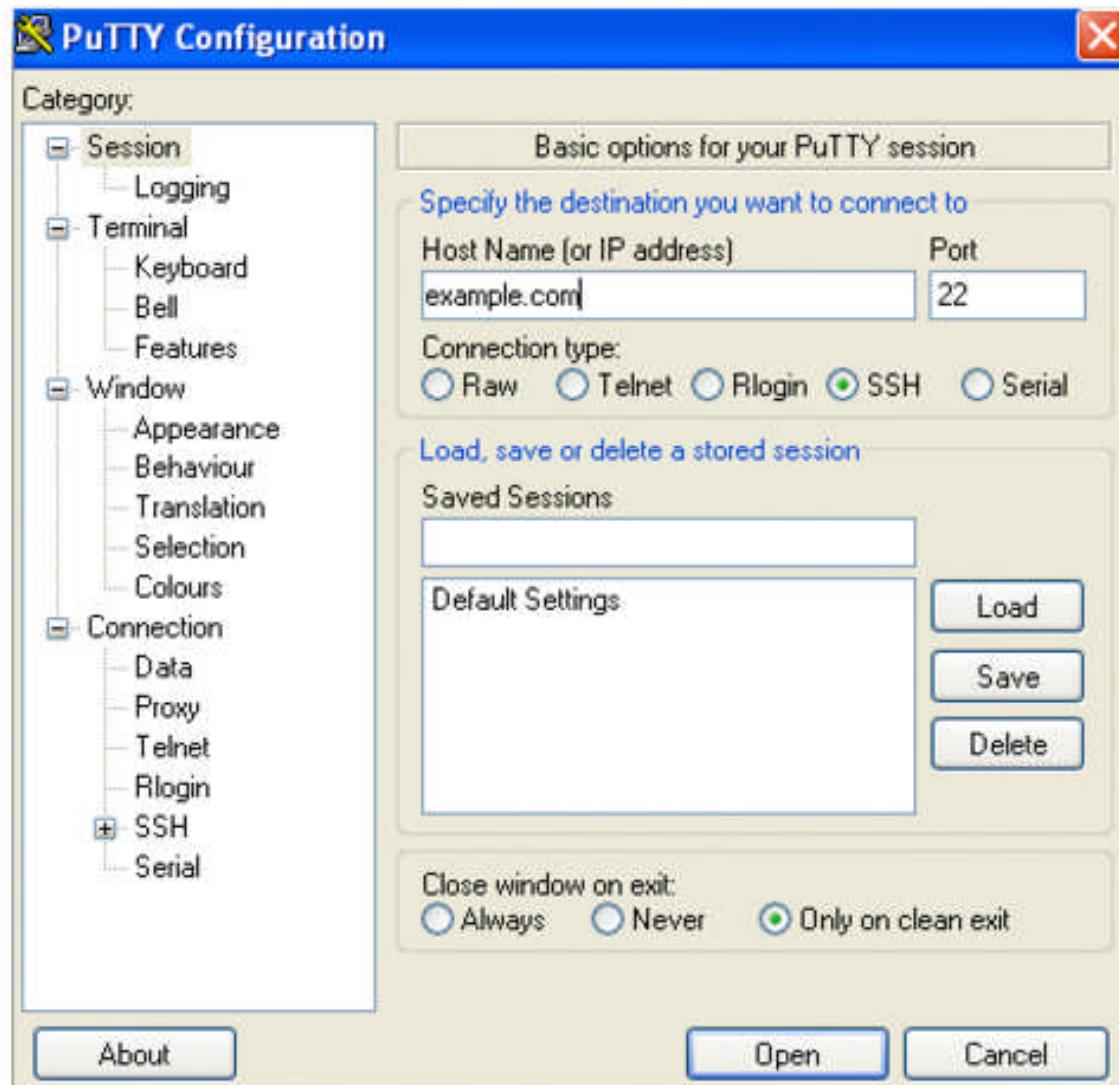
putty-prv.ppk .. putty keypair

(prvkey kfile.ppk is locked with passphrase)

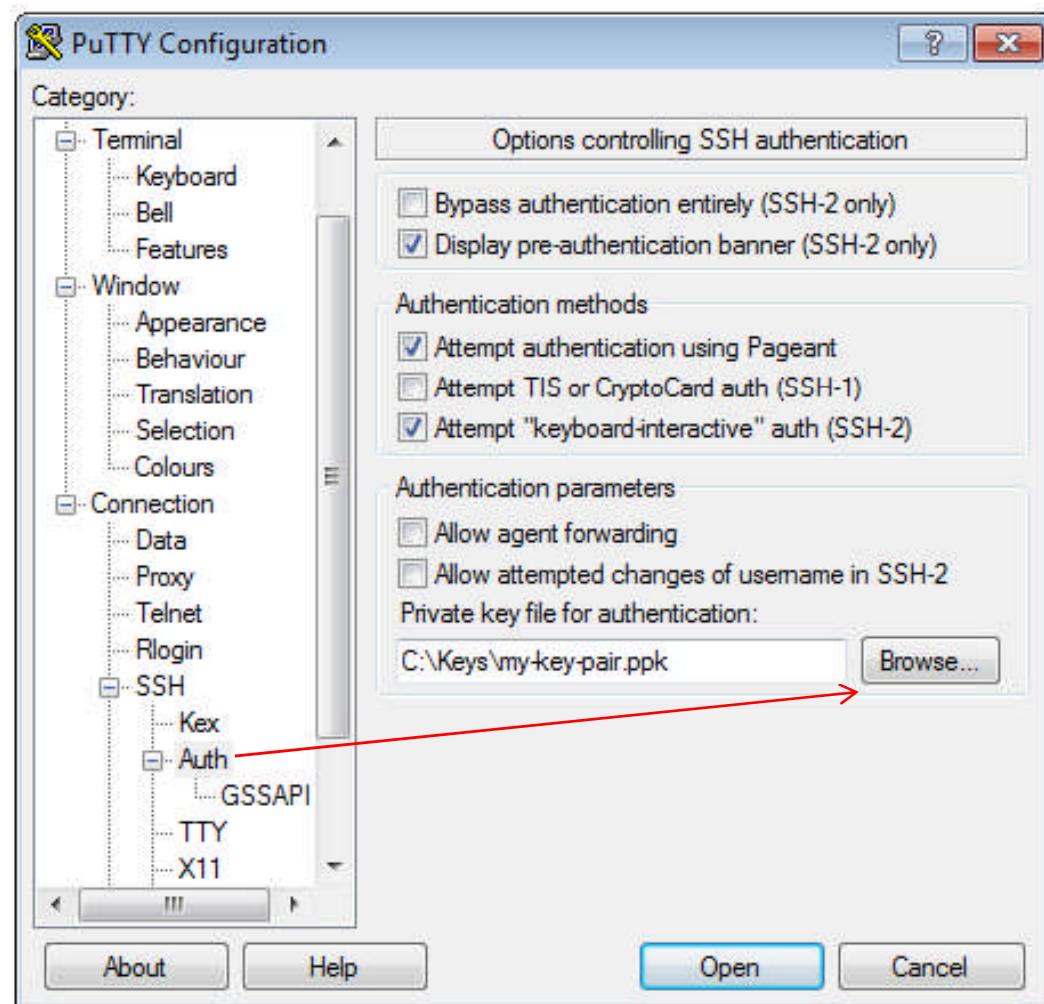
known\_hosts .. ssh list of hosts

authorized\_keys .. list of pubkeys allowed to login.

# Using Putty for remote login



# Add keypair.ppk to putty



# Distributing PublicKey to server

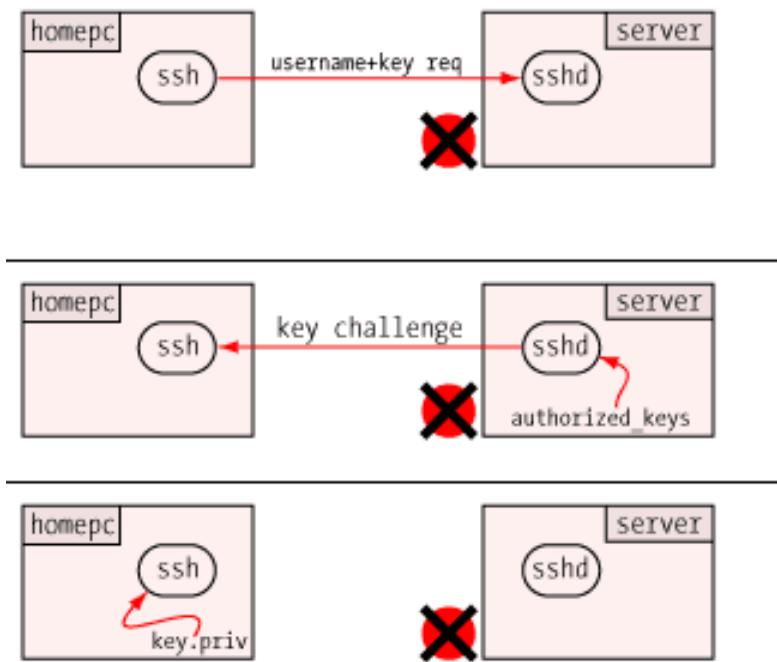
On the Server paste your PublicKey in  
`~/.ssh/authorized_keys`

```
$ cat ~/.ssh/id_rsa.pub >>
~/.ssh/authorized_keys
```

CPANEL: upload public key and authorize it.

# SSH Login 1

- 1 The user makes an initial connection and sends a username along with a request to use a key.
- 2 The server's sshd looks in the user's authorized\_keys file, constructs a challenge based on the public key found there, and sends this challenge back to the user's ssh client.
- 3 The ssh client receives the key challenge. It finds the user's private key on the local system, but it's protected by an encrypting passphrase. An RSA key file is named id\_rsa on OpenSSH and SecureCRT, keyname.ppk on PuTTY. Other types of keys (DSA, for instance) have similar name formats.



from <http://www.unixwiz.net/techtips/ssh-agent-forwarding.html>

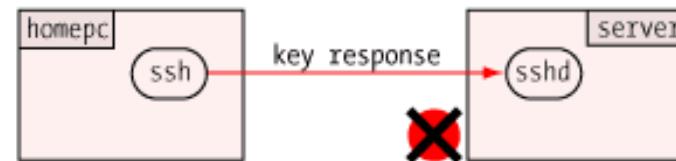
# SSH Login 2

4 The user is prompted for the passphrase to unlock the private key. This example is from [PuTTY](#).

5 ssh uses the private key to construct a key response, and sends it to the waiting sshd on the other end of the connection. It does **not** send the private key itself!

6 sshd validates the key response, and if valid, grants access to the system.

```
linux - PuTTY
Using username "steve".
Authenticating with public key "steve@unixwiz.net"
Passphrase for key "steve@unixwiz.net": █
```



# Keymanager: pageant

> start pageant keyfile.ppk

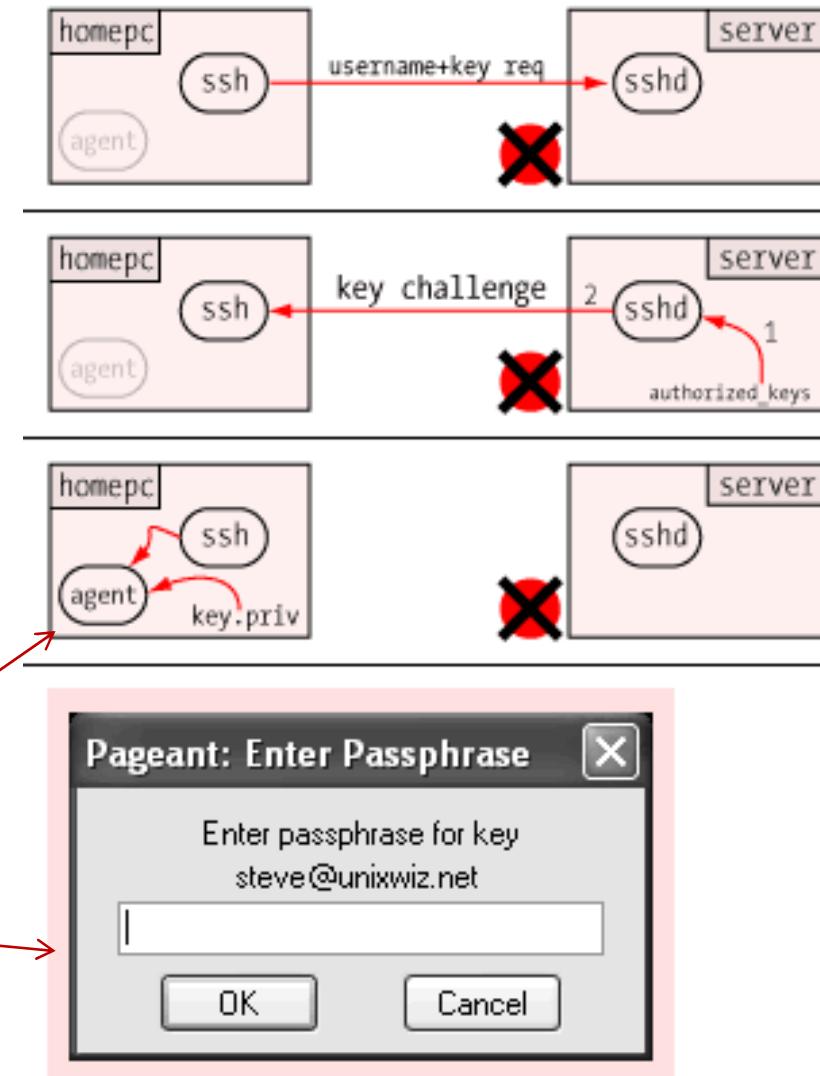
your pass phrase: \*\*\*\*

> putty will automatically get your keyfile  
from the keymanager pageant, and no  
need to type passphrase in putty at all.

# SSH Login with keymanager 1

- 1 The user makes an initial connection and sends a username along with a request to use a key.
- 2 The ssh daemon on the server looks [1] in the user's **authorized\_keys** file, constructs a challenge based on the key, and sends it [2] back to the user's ssh client.
- 3 The ssh client receives the key challenge, and forwards it to the waiting agent. The agent, rather than ssh itself, opens the user's private key and discovers that it's protected by a passphrase.
- 4 The user is prompted for the passphrase to unlock the private key.

This example shows the prompt from [PuTTY's pageant](#).



# SSH Login with keymanager 2

5 The agent constructs the key response and hands it back [1] to the **ssh** process, which sends it off [2] to the **sshd** waiting on the other end. Unlike the previous example, **ssh never sees the private key directly, only the key response.**



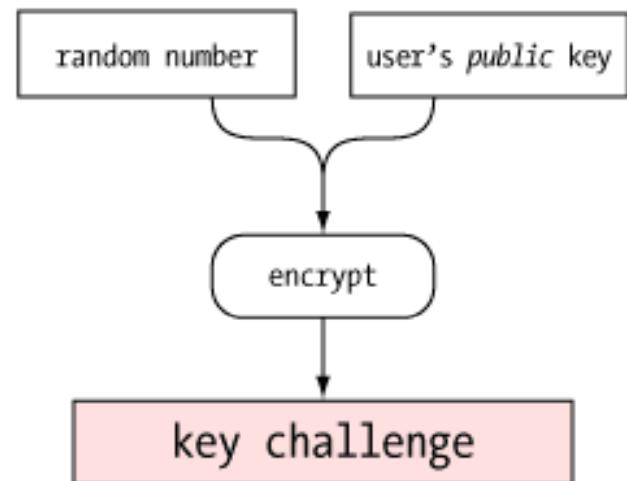
6 **sshd** validates the key response, and if valid, grants access to the system. Note: the agent still retains the private keys in memory, though it's not participating in the ongoing conversation.



# Key Challenge 1

1. How it can verify a user's identity (or more precisely, possession of a private key) without revealing that private key to anybody?
2. When a user wishes access to an ssh server, he presents his username to the server with a request to set up a key session. This username helps locate the list of public keys allowed access to that server, found in the `~/.ssh/authorized_keys` file.
3. The server creates a "challenge" which can only be answered by one in possession of the corresponding user's private key. Server creates and remembers a large random number, then encrypts it with the user's public key. This blob is sent as a challenge to the user requesting access.

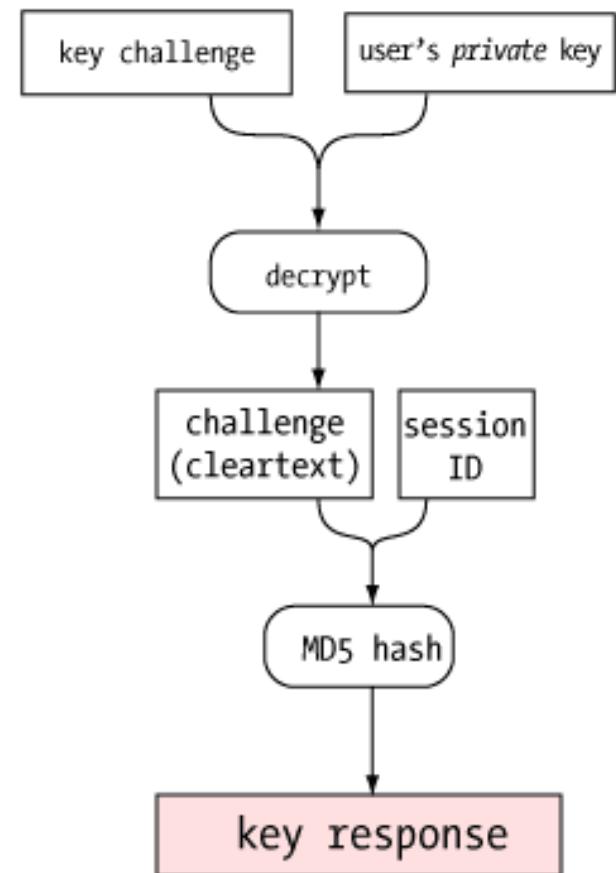
Key Challenge Creation



# Response to Challenge 2

1. User receives the challenge, and decrypts it with the private key, revealing the original random number from the server. [Only the holder of the private key could ever extract this random number, so this constitutes proof that the user is the holder of the private key].
2. The agent takes this random number, appends the SSH session ID (which varies from connection to connection), and creates an MD5 hash value of the resultant string: this result is sent back to the server as the **key response**.
3. The server computes the same MD5 hash (random number + session ID) and compares it with the user response. If they match, the user has the private key, and access is granted.

Key Response Generation



# Using ssh command line

> ssh -i KEYPAIR SERVER COMMAND

> ssh me@myserver.com ls /home/me  
[default keypair is ~/.ssh/id\_rsa]

> c:/cygwin/bin/rsync -e 'ssh -i mykey' \  
--list-only me@myserver.com:/home/me

# bitbucket: git over ssh

```
C:\bitbucket> cat ~/.ssh/id_rsa.pub | clipboard
```

Pasted into <https://bitbucket.org/account/user/NAME/ssh-keys/>

```
c:\bitbucket> cat ~/.ssh/config
```

Host bb

user yourid

hostname bitbucket.org

IdentityFile ~/.ssh/id\_rsa

```
C:\bitbucket> ssh-agent /bin/bash
```

```
sh-4.3$ ssh-add ~/.ssh/id_rsa
```

```
Enter passphrase .. id_rsa: ****
```

OK

```
> git clone git@bitbucket.org:YOUR_PROJ.git
```

OK

# git on windows

Tell Windows git to use your keys,

```
> set GIT_SSH_COMMAND=ssh -i /path/id_rsa
```

Don't use quotes in set command

```
> set GIT_SSH_COMMAND="ssh -i /path/id_rsa" .. WRONG
```

# Using scp

Exampe 1. Copy Local file to remote server

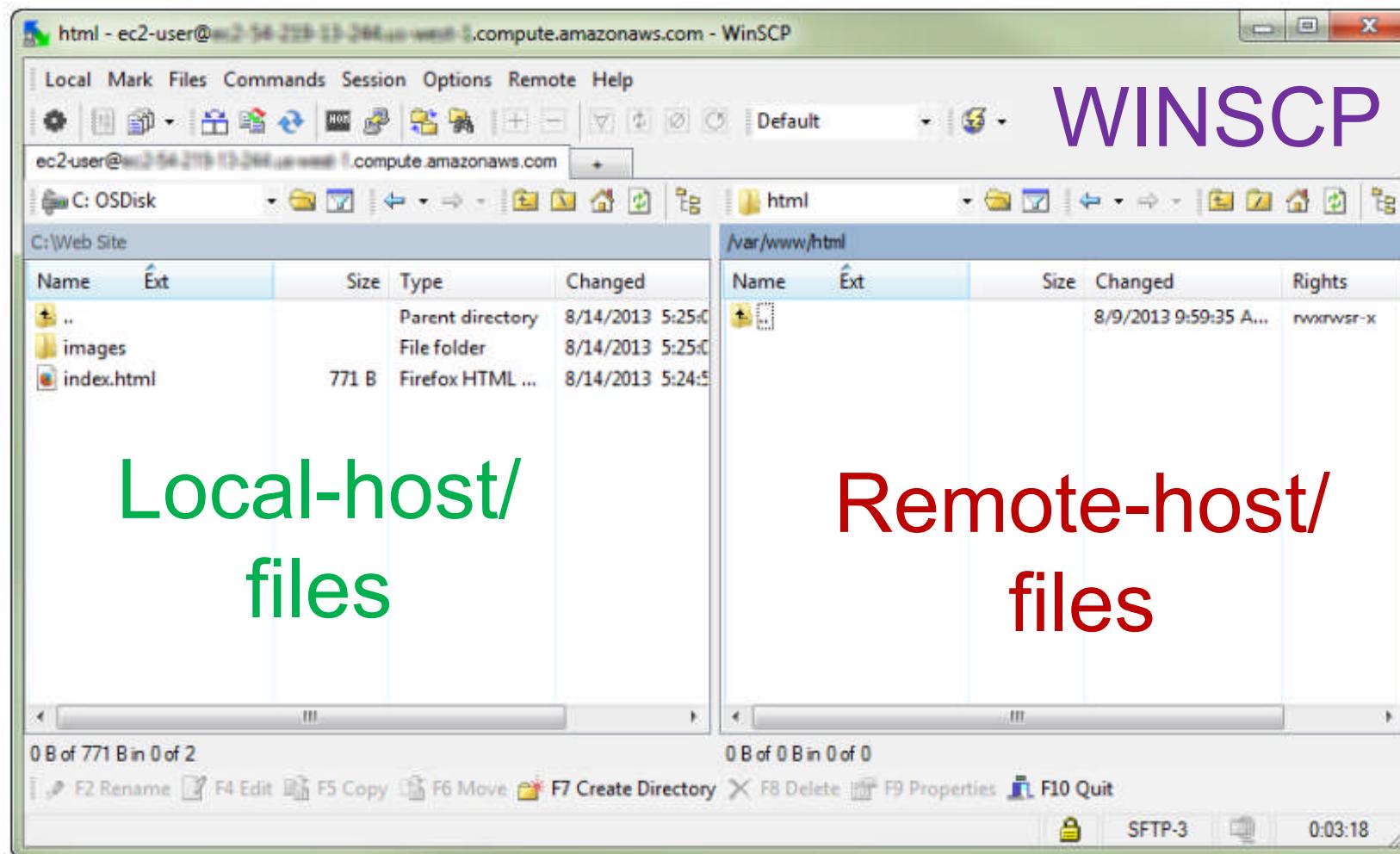
```
> scp -i KEYPAIR LOCALPATH USER@SERVER:/REMOTE/PATH
> scp -i mykey file.txt me@myserver.com:/home/me/file.txt
```

Example 2. Copy file between two remote servers:

```
$ cat ~/.ssh/config
Host remote1.example.org
Port 2222
IdentityFile /path/to/host1-id_rsa
Host remote2.example.org
Port 6969
IdentityFile /path/to/host2-id_rsa
$ scp -3 user1@remote1:/home/user1/file1.txt \
user2@remote2:/home/user2/file1.txt
```

# Using winscp to copy files

(winscp uses same ppk keys)



# References

1. [https://en.wikipedia.org/wiki/Public-key\\_cryptography](https://en.wikipedia.org/wiki/Public-key_cryptography)
2. <http://www.unixwiz.net/techtips/ssh-agent-forwarding.html>

# Using GPG

1. Making your public key
2. Signing

# GPG (Gnu privacy guard)

# PGP (pretty good privacy)

- GPG
- C:\> gpg --version
- C:\> gpg --gen-key
  - » gpg: key 43F2B829 marked as ultimately trusted
  - » public and secret key created and signed.
- ~/.gnupg
- gpg --export --armor > Public-key.asc
- gpg --import file.asc
- gpg --sign-key RedHat

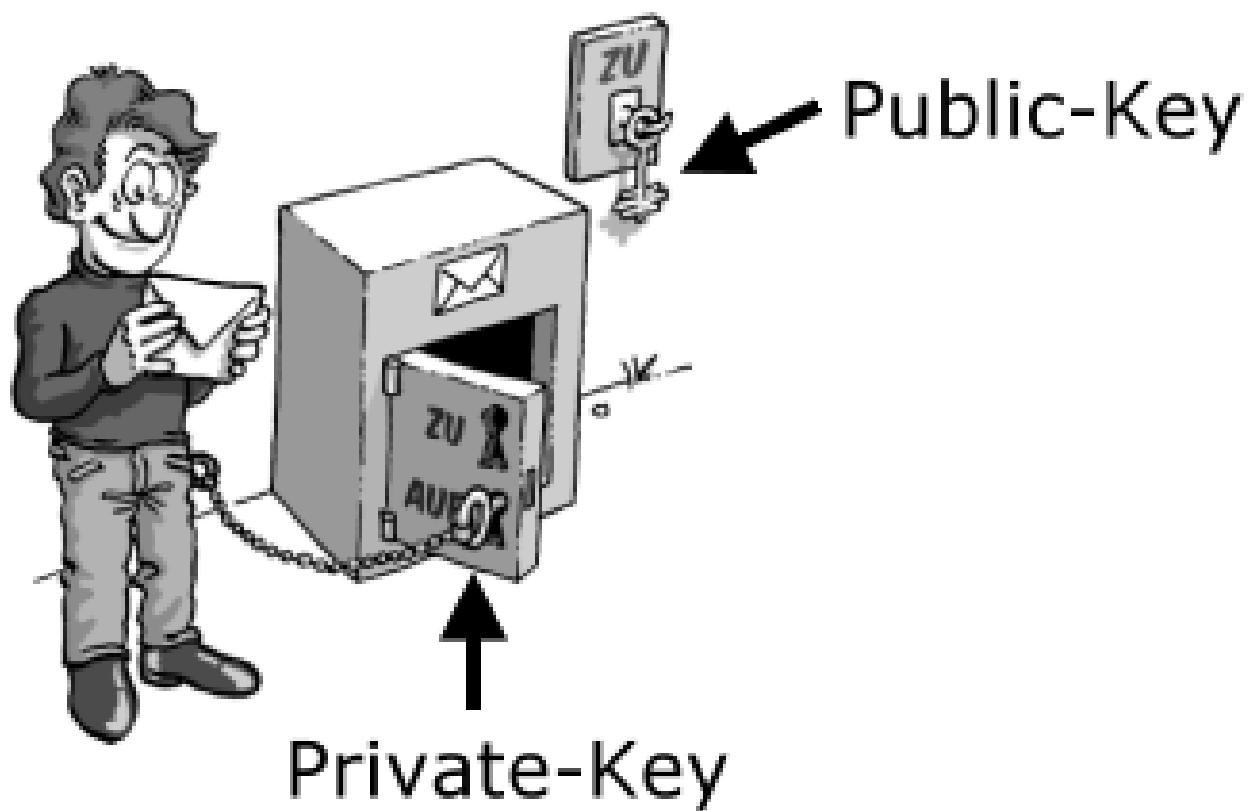
# Password length and strength

## **Completely random printable string**

SIZE, BITS, CRACK-TIME

- 6 char, 40-bits, Minutes
- 8 char, 52-bits, Hours
- 12 char, 78-bits, Decades
- 15 char, 97-bits, Centuries
- 20 char, 130-bits, Un-crackable

# Key pair



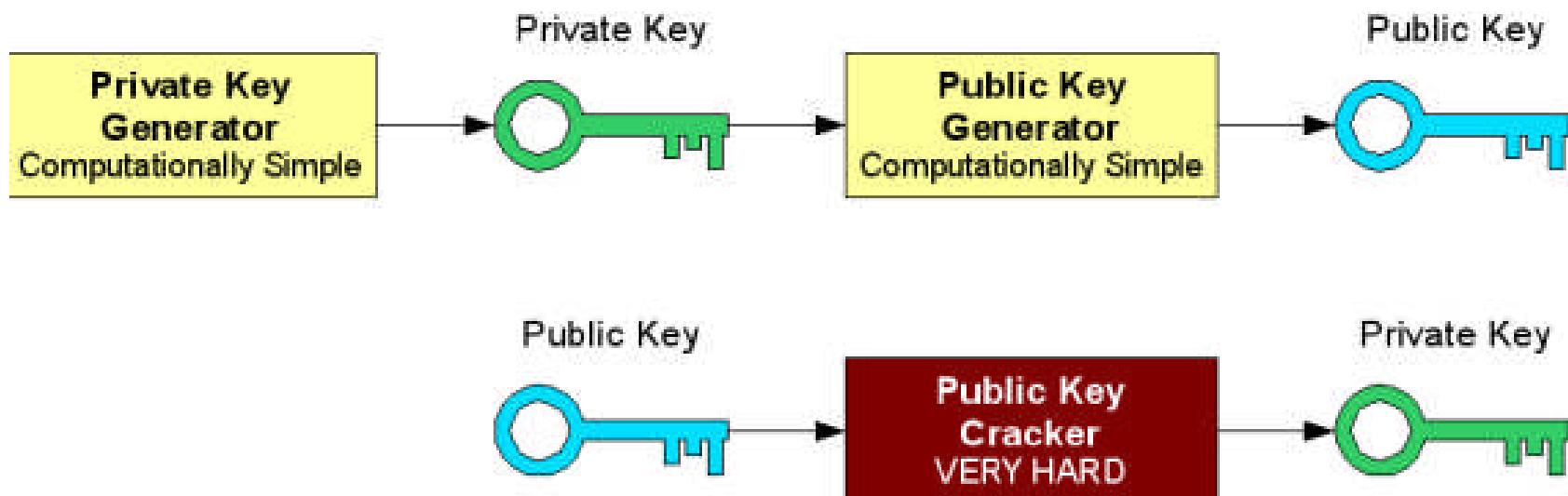
# Passphrase protects your keypair

Private Keys

Pass Phrase



# KeyPair is Easy to Generate, but Hard to factor



# Making your own public key

GPG : GNU Privacy Guard

1. MS-Windows install gpg4win (GUI)  
cygwin has gpg commandline
2. Linux install gpg if needed.

Online help google for gnupg

<https://help.ubuntu.com/community/GnuPrivacyGuardHowto>

<http://www.gnupg.org/gph/en/manual.html>

# windows Gpg4Win

Incase your unix \$HOME is different from windows %HOME%, then link them together with this command:

```
c:\> mklink /D %APPDATA%\gnupg %HOME%\gnupg
```

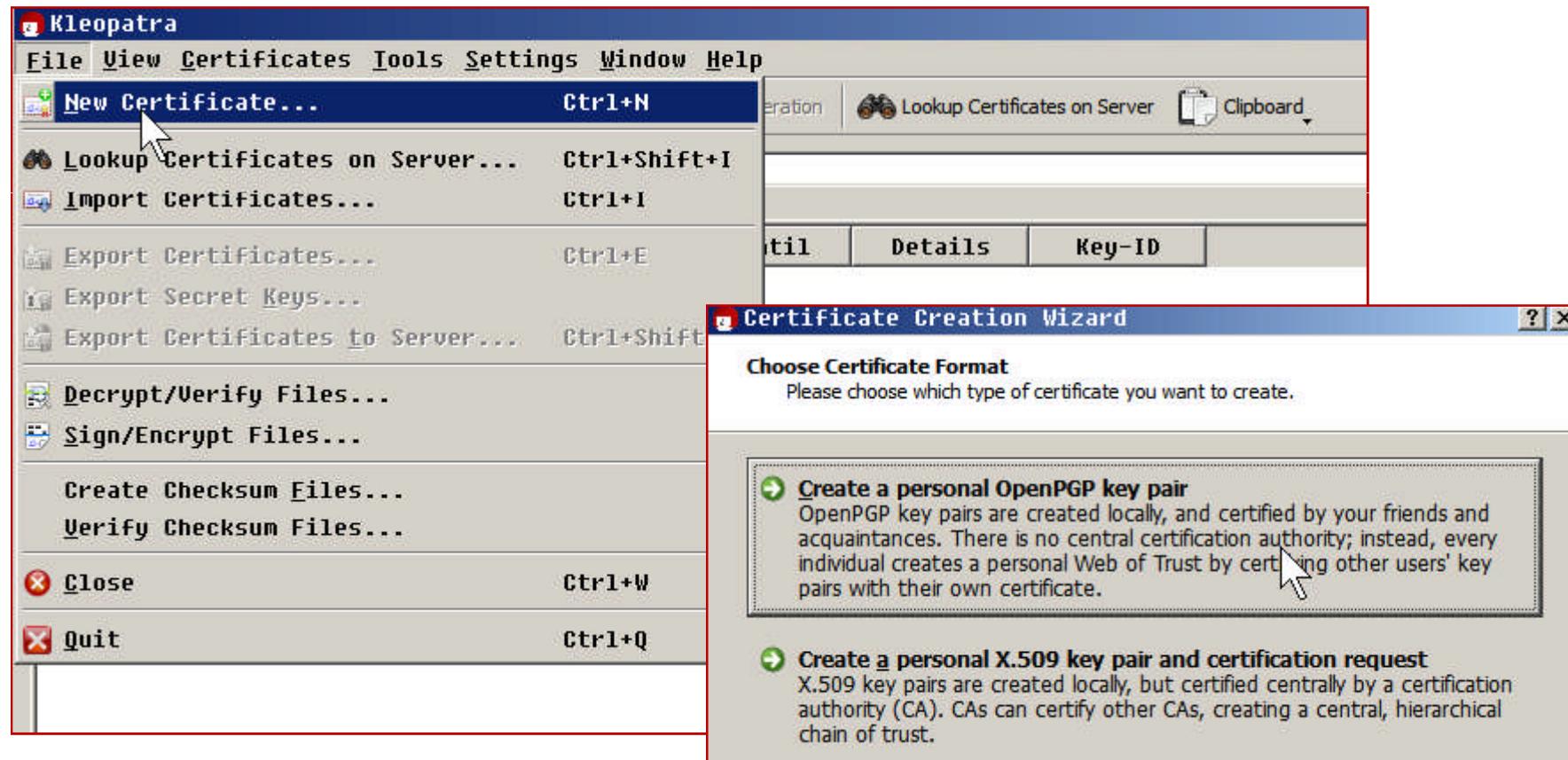
Where APPDATA is C:\Users\%USERNAME%\AppData\Roaming

# keymanager

Keymanager GUI is called kleopatra

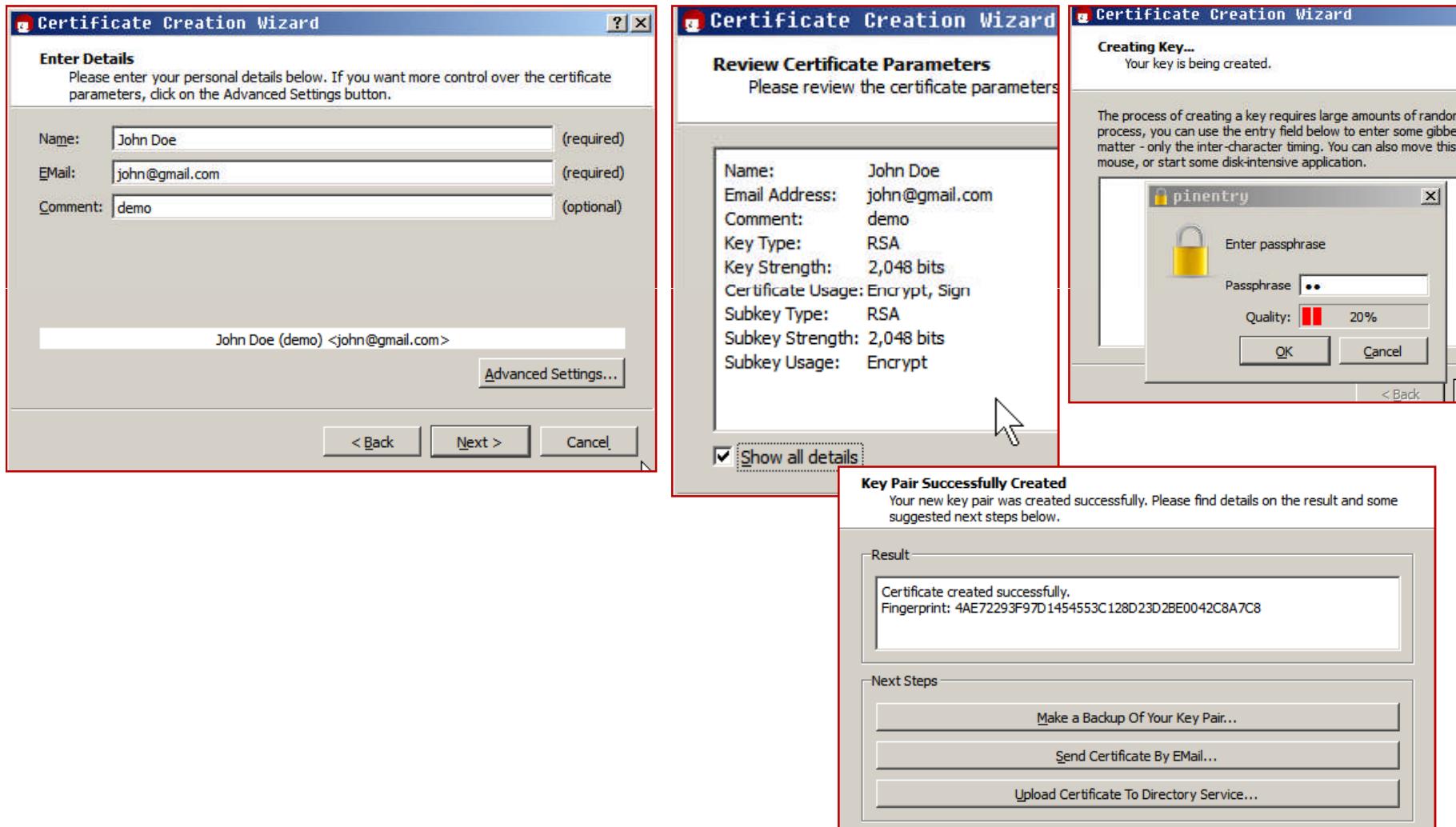
c:\> start c:\tools\gpg4win\kleopatra.exe

File > New Certificate > Create Personal OpenPGP key pair

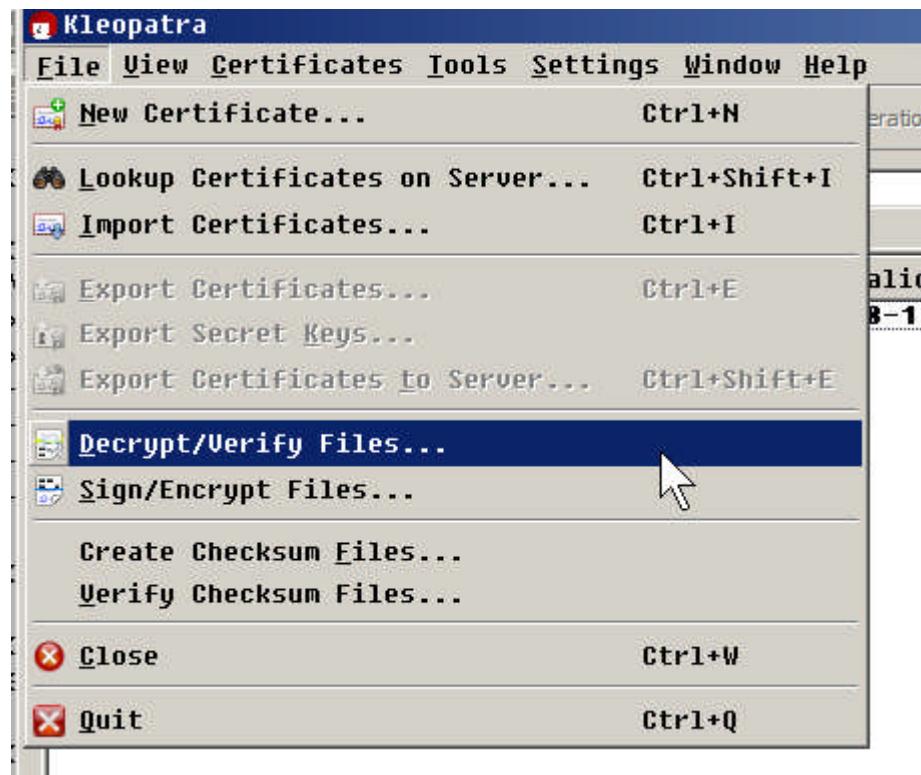


# make a key

Enter your name, Email > next > passphrase > created



# Ready to use key in Kleopatra



# GPG usage 2

- Generate a private key:

```
gpg --gen-key
```

- Get your public key as ascii text:

```
gpg --armor --output pubkey.txt --export you@ashesi.edu.gh
```

- Send your keys to a key-server

```
gpg --send-keys youremail --keyserver hkp://subkeys.pgp.net
```

- Import Friend's key

```
gpg --import friend.asc OR gpg --search-keys 'friend@ashesi.edu.gh'
--keyserver hkp://subkeys.pgp.net
```

See <http://www.gnupg.org/gph/en/manual.html> for more help.

# Using GnuPG to generate your keys

- Open a [terminal](#) and enter:  
  \$ gpg --gen-key
- Please select what kind of key you want:
  - (1) RSA and RSA (default)
  - (2) DSA and Elgamal
  - (3) DSA (sign only)
  - (4) RSA (sign only)
- Your selection? **1**
  - RSA keys may be between 1024 and 4096 bits long.
- What keysize do you want? (**2048**)
  - Requested keysize is 2048 bits
- Please specify how long the key should be valid.
  - 0 = key does not expire
  - <n> = key expires in n days
  - <n>w = key expires in n weeks
  - <n>m = key expires in n months
  - <n>y = key expires in n years
- Key is valid for? (**0**)
  - Key does not expire at all
- Is this correct? (y/N) **y**

# command line key generation

- You need a user ID to identify your key; the software constructs the user ID
- Real name: **Mohsin Ahmed**
- Email address: **moshahmed@gmail.com**
- Comment: NITK demo
- You selected this USER-ID:
  - "Mohsin Ahmed (NITK demo) <moshahmed@gmail.com>"
- Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? **O**
  - You need a Passphrase to protect your secret key.
- Passphrase: **\*\*\*** (e.g. a short sentence)
- Repeat Passphrase: **\*\*\***
  - Forgetting your passphrase will result in your key being useless.  
Carefully memorize your passphrase.

# Key pair is created

- ..+++++
- .....++++
- gpg: /cygdrive/c/mosh/.gnupg/trustdb.gpg: trustdb created
- gpg: key B30E2394 marked as ultimately trusted
- public and secret key created and signed.
  
- gpg: checking the trustdb
- gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
- gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
- pub 2048R/**B30E2394** 2013-02-09 ([your public key](#))
- Key fingerprint = FB38 60F1 DAD5 0F64 A8E4 465D 249B  
A4E8 B30E 2394
- uid Mohsin Ahmed (NITK demo) <[moshahmed@gmail.com](mailto:moshahmed@gmail.com)>
- sub 2048R/3B0695DE 2013-02-09

# Submit your public key to Key server

e.g. <http://pgp.mit.edu/>

## MIT PGP Public Key Server

Help: [Extracting keys](#) / [Submitting keys](#) / [Email interface](#) / [About this server](#) / [FAQ](#)

Related Info: [Information about PGP](#) / [MIT distribution site for PGP](#)

---

### Extract a key

Search String:  Do the search!

Index:  Verbose Index:

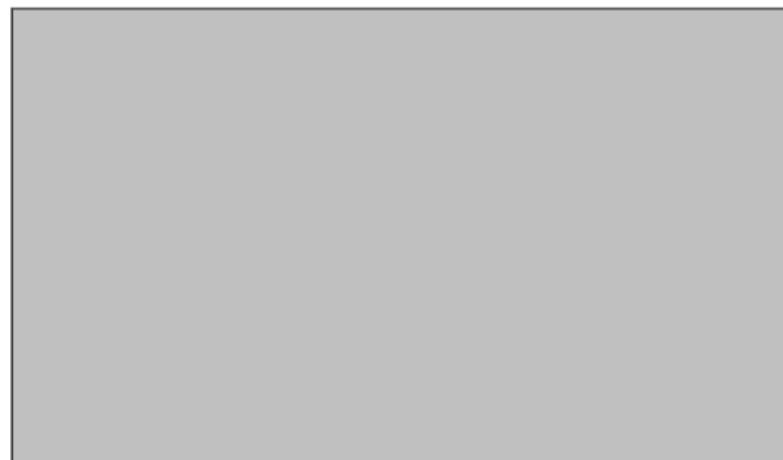
Show PGP fingerprints for keys

Only return exact matches

---

### Submit a key

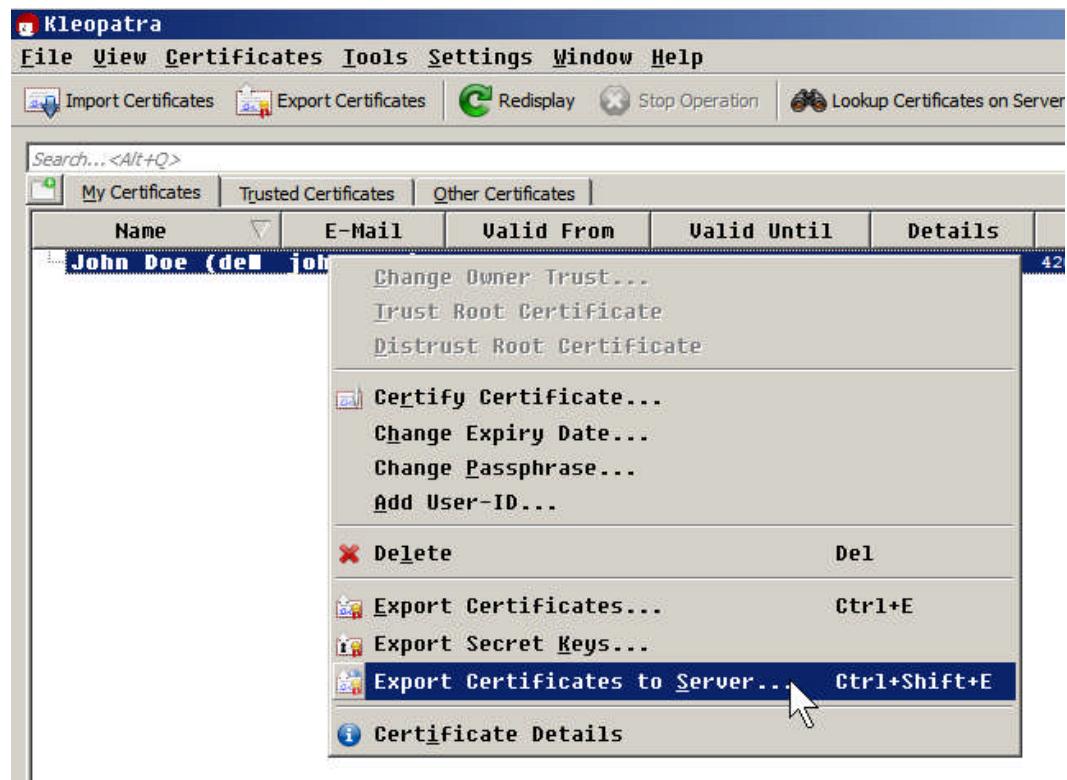
Enter ASCII-armored PGP key here:



# Upload your public key

\$ gpg --keyserver [pgp.mit.edu](http://pgp.mit.edu) --send-key B30E2394

OR use kleopatra



# Print your key

```
$ gpg -v --fingerprint B30E2394
```

- pub 2048R/B30E2394 2013-02-09
- Key **fingerprint = FB38 60F1 DAD5 ...**
- uid Mohsin Ahmed (NITK demo) [moshahmed@gmail.com](mailto:moshahmed@gmail.com)
- Print the fingerprint, and give the printout to your friends (who know you), ask them to sign it and certify it as your key.
- You will do the same for people you know well.

# GPG usage 3

- Encrypt message.txt for your friend:

```
gpg --encrypt --recipient friend@ashesi.edu.gh
message.txt
```

- Reading mail from your friend

```
gpg --decrypt reply.txt
```

- Signing a file

```
gpg --armor --detach-sign my-file.zip
```

- Verify the sign

```
gpg --verify crucial.zip.asc crucial.zip
```

# Signing someone's key

- Download their key (e.g. 00AA11BB) or from their email (verify their fingerprint).

```
$ gpg --keyserver pgp.mit.edu --recv-keys
00AA11BB
```

Check their fingerprint:

```
$ gpg --fingerprint 00AA11BB
```

Sign it:

```
$ gpg --sign-key 00AA11BB
```

# Send them their signed certificate

```
$ gpg --armor --output cert1.txt --export
00AA11BB
$ mail friend < cert1.txt
```

# Update your signed keys

When you receive your cert2.txt signed from  
a friend:

```
$ gpg --import cert2.txt
```

Update your key in the server:

```
$ gpg --keyserver pgp.mit.edu --send-key
B30E2394
```

# Send someone a locked file

```
$ gpg --output locked.txt --encrypt
--recipient friend@gmail.com poem.txt
```

Now mail the file ‘locked.txt’ to friend.

Friend can read the attached file with the command:

```
$ gpg --output poem.txt --decrypt locked.txt
```

# Sign a cheque

Create cheque.txt and then sign it:

```
$ gpg --output check.sig --clearsign
check.txt
```

Mail the signed check.sig to your friend.

Friend can verify your signature:

```
$ gpg --verify check.sig
```

# Sign a cheque

Create cheque.txt and then sign it:

```
$ gpg --output check.sig --clearsign
check.txt
```

Mail the signed check.sig to your friend.

Friend can verify your signature:

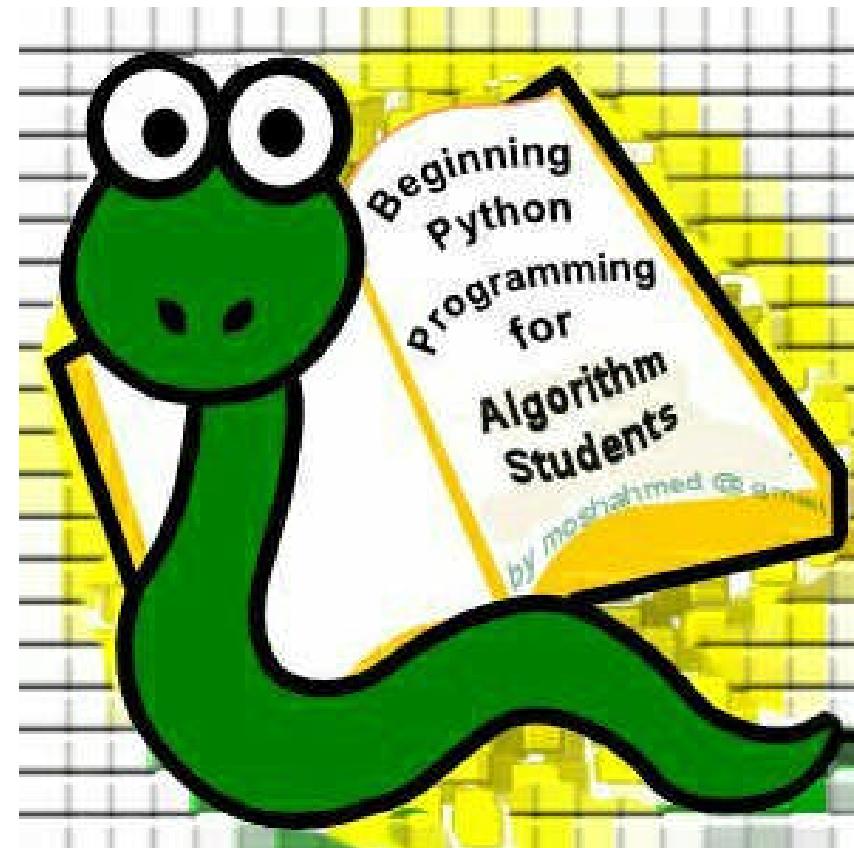
```
$ gpg --verify check.sig
```

# Key manager: gpg-agent

- Instead of having to enter the passphrase each time, run a key manager:
- Linux: Start gpg-agent, give it your passphrase, and it will provide your privatekey to applications as needed.
- Windows: use kleopatra in Gpg4Win
- For putty keys: start pageant keyfile.ppk

# Python

- Python is a scripting language with lots of libraries
- Classes and objects.
- Indent to organize code.



# Python

- Useful for small calculations, scripts
- Install python33 / python27 / numpy superpack
- Usage:

```
C:\> C:\python33\python.exe
```

```
>>> 2+2
```

```
4
```

```
>>> ^Z # (^D on unix) EOF to exit
```

```
C:\> # back to DOS prompt
```

# Python calculator

```
C:\> c:\python33\python.exe
>>> import math
>>> dir(math) # see what's in math ...
>>> math.pi # math object has attribute pi with value=...
3.141592653589793
>>> math.log2(math.pi) # math object has member function log2
1.6514961294723187

>>> from math import * # Make all math functions are global
>>> sin(pi/4)
0.70710678118654746
>>> log2(pi)
1.6514961294723187
```

# Function example: quicksort

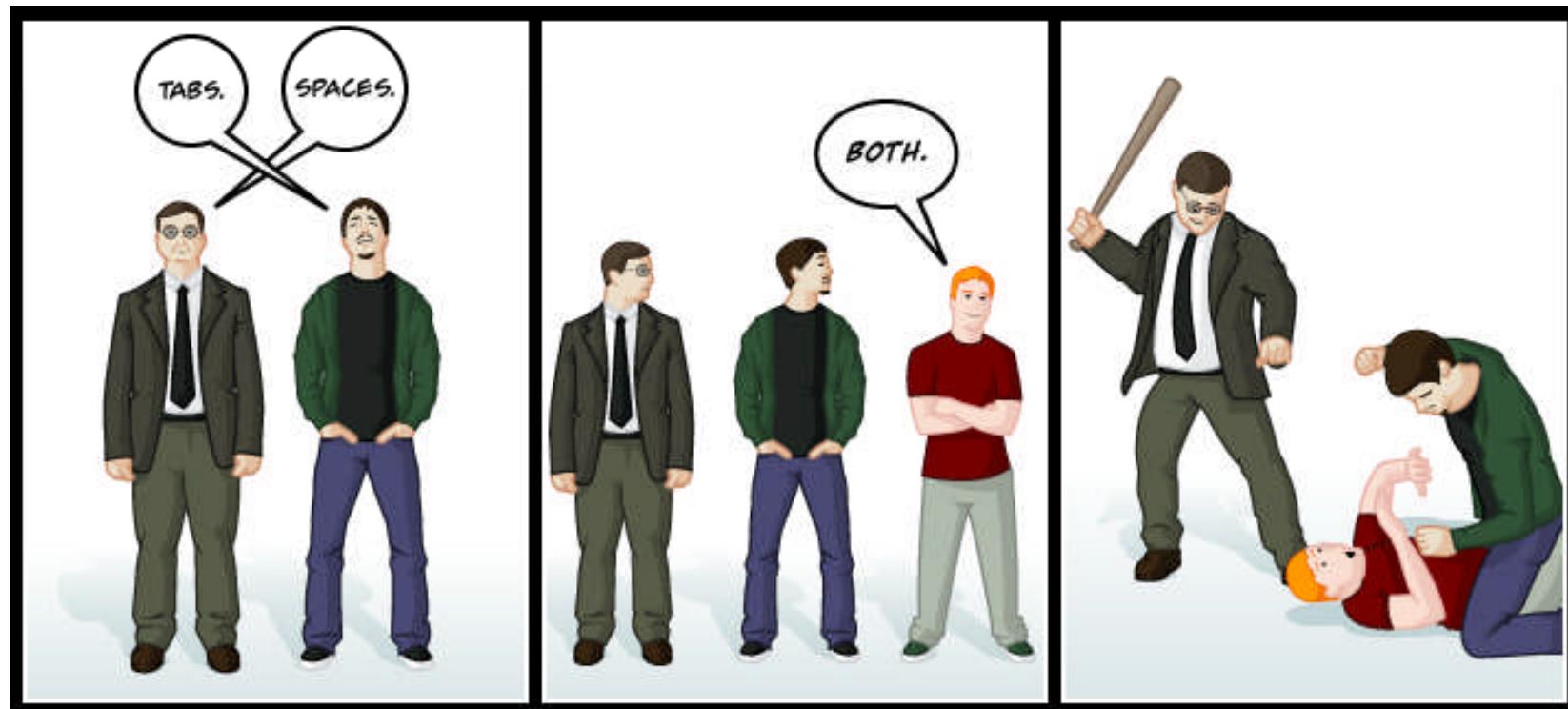
```
>>> def quicksort(arr):
... """quicksort in python3"""\n... # Triple quoted string.
... if len(arr) <= 1: return arr
... pivot = arr[0]
... return quicksort([x for x in arr[1:] if x < pivot]) + [pivot] + \
... quicksort([x for x in arr[1:] if x > pivot])
...
...>>> print(quicksort([4,5,1,3,2]))
```

```
[1, 2, 3, 4, 5]
```

- '...' means python is prompting for more input

# Don't use tabs,

- 1.Indent with 4 spaces
2. Edit with vim, `~/.vimrc: set ts=4 sw=4 et`



# Linear Algebra using Python

- Install Python
- Install NumPy superpack python library

# Python Linear Algebra calculator

```
C:\> python
>>> import numpy as np
>>> x = np.array(((2, 3),
 (3, 5)))
>>> y = np.matrix(((1, 2),
 (5, -1)))
>>> np.dot(x,y)
matrix([[17, 1], # 17=2*1+3*5, 1=2*2-3*1
 [28, 1]]) # 28=3*1+5*3, 1=3*2=5*1
```

# Python solver and graphics

**Solve** the system of equations

$$\begin{aligned}3 * x + 1 * y &= 9 \\1 * x + 2 * y &= 8\end{aligned}$$

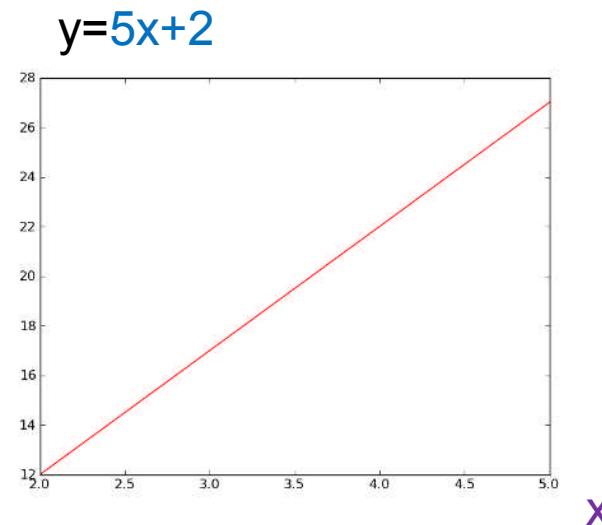
```
C:\> python
>>> import numpy as np
>>> a = np.array([[3,1], [1,2]])
>>> b = np.array([9,8])
>>> x = np.linalg.solve(a, b)
>>> x
array([2., 3.]) # solution: x=2, y=3
```

```
>>> (np.dot(a, x) == b).all()
```

True

---

```
import matplotlib.pyplot as plt
>>> plt.plot(x, 5*x + 2, 'r')
>>> plt.show()
```



# Matrix inverse

```
1. >>> import numpy as np
2. >>> from scipy import linalg
3. >>> A = np.mat('1 3 5;
 2 5 1;
 2 3 8]')
1. >>> linalg.inv(A)
2. array([-1.48, 0.36, 0.88],
3. [0.56, 0.08, -0.36],
4. [0.16, -0.12, 0.04]])
5. >>> linalg.det(A)
6. -25.
```

When Matrix  $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Formula for  
2x2 Matrix Inverse

# Exercise

Compute inverse of A using python

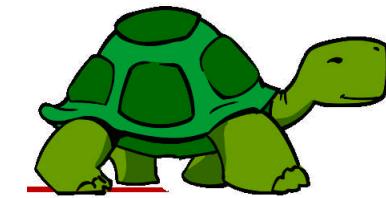
```
>>> A = [[3, 4],
 [2, 3]]
```

```
>>>
```

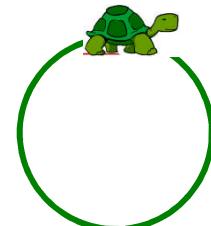
# Solution

```
>>> from numpy import *
>>> A = [[3, 4], [2, 3]]
>>> B = linalg.inv(A)
array([[3., -4.],[-2., 3.]])
>>> dot(A,B) # Not A*B
array([[1., 0.], [0., 1.]])
```

# Python Turtle Graphics



```
C:\> c:\python33\python.exe
>>> import turtle # load a library
>>> dir(turtle) # lists members of turtle
>>> turtle.circle(100) # draws a circle
```

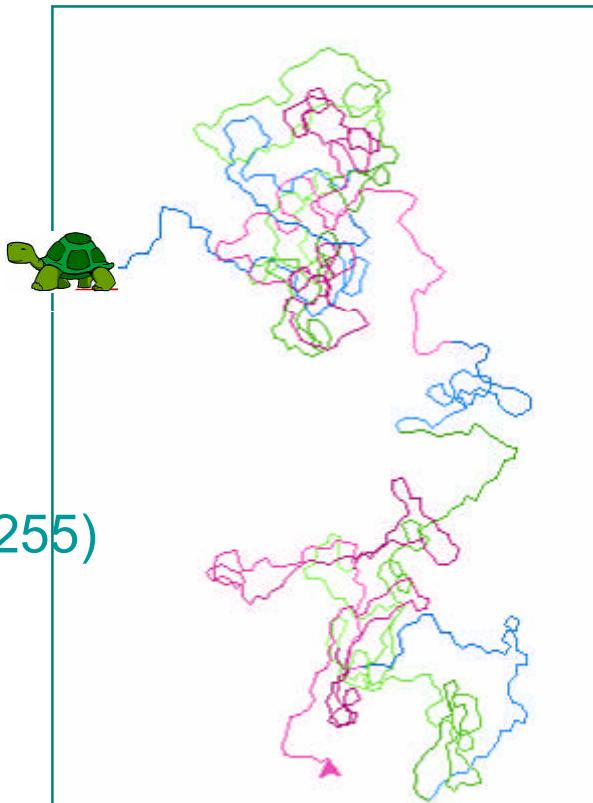


# Turtle Graphics in Python3

C:\> cat random-walk.py

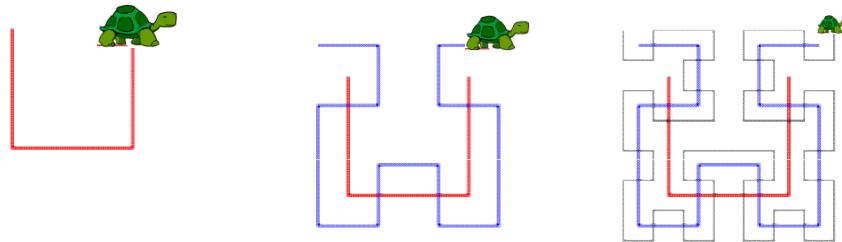
```
1. #!python3
2. import turtle, random
3. wn = turtle.Screen()
4. turtle.colormode(255)
5. turtle.speed(0)
6. for i in range(1000):
7. turtle.forward(5)
8. turtle.left(random.random() * 180 - 90)
9. turtle.color(i%255,(i+100)%255,(i+200)%255)
10. wn.mainloop()
```

C:\> c:\python33\python.exe random-walk.py

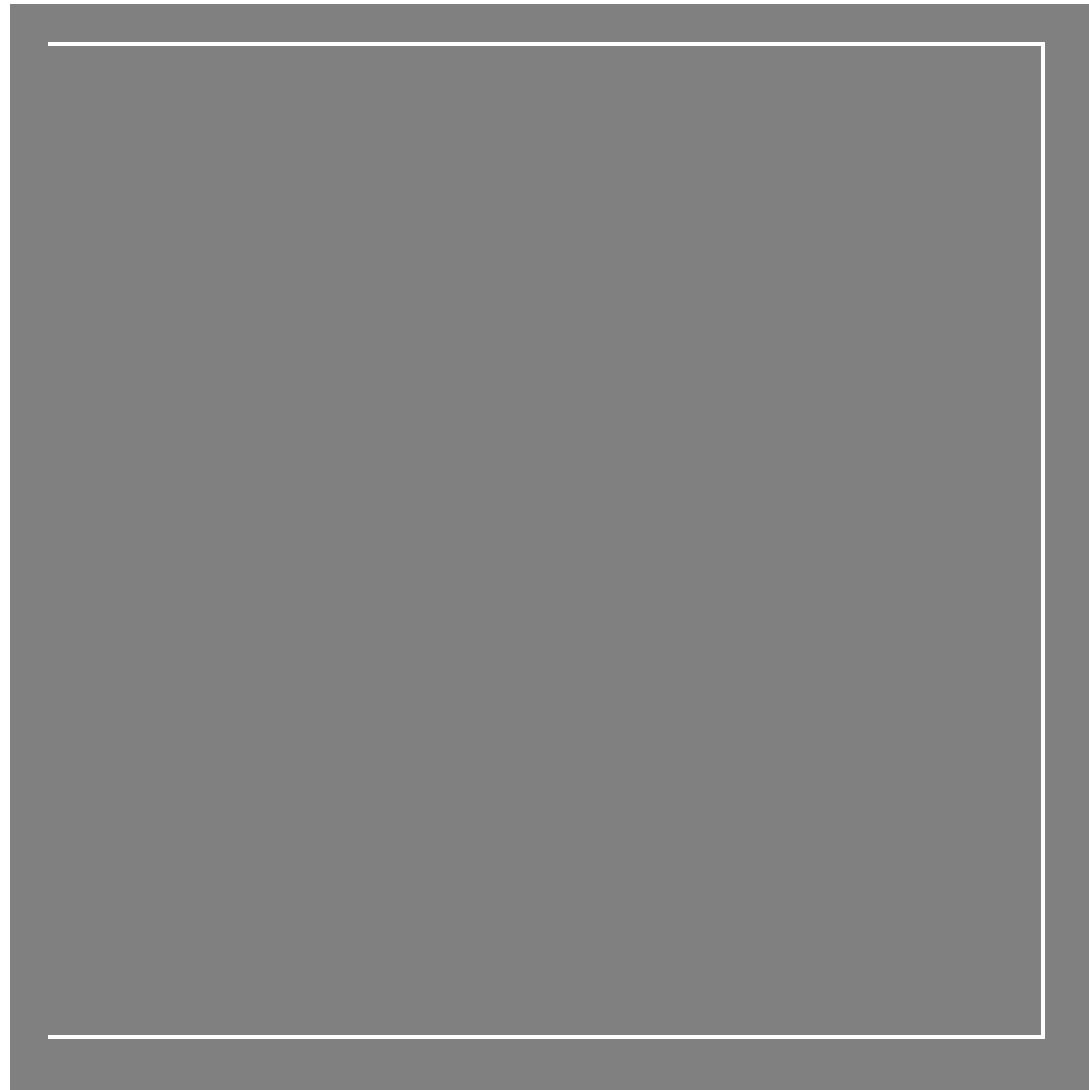


# Hilbert space-filling curve

C:\> c:\python33\python.exe hilbert.py

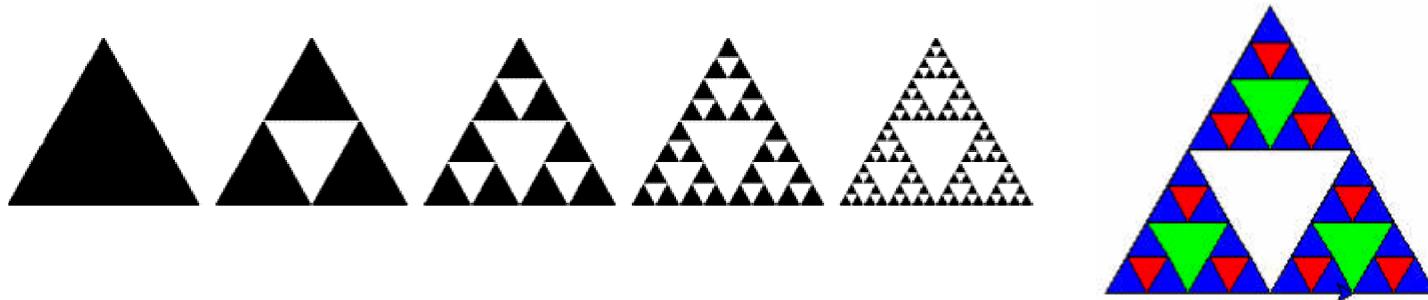


# Animated gif of hilbert curve

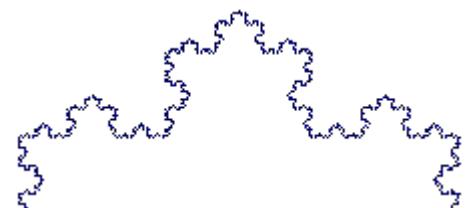


# Sierpinski triangles

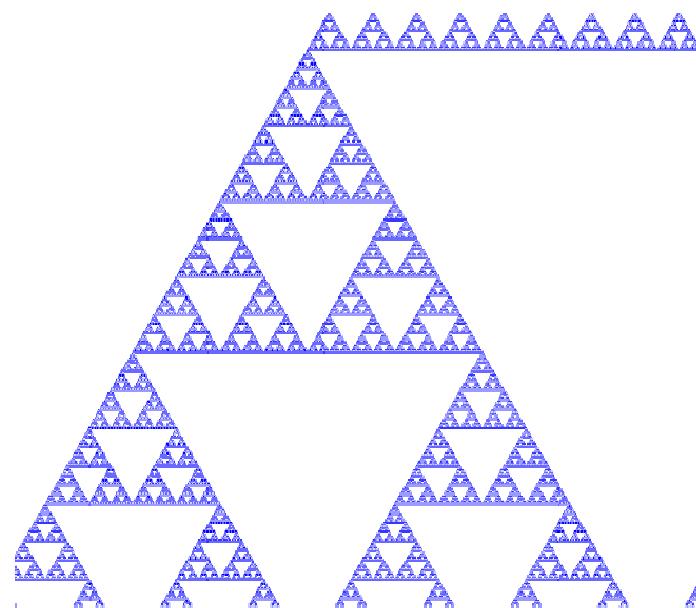
- Fractals: Originally constructed as a curve, this is one of the basic examples of self-similar sets, i.e. it is a mathematically generated pattern that can be reproducible at any magnification or reduction.
- The Sierpinsky Triangle is a fractal created by taking a triangle, decreasing the height and width by 1/2, creating 3 copies of the resulting triangle, and place them such each triangle touches the other two on a corner. This process is repeated over and over again with the resulting triangles to produce the **Sierpinski** triangle:



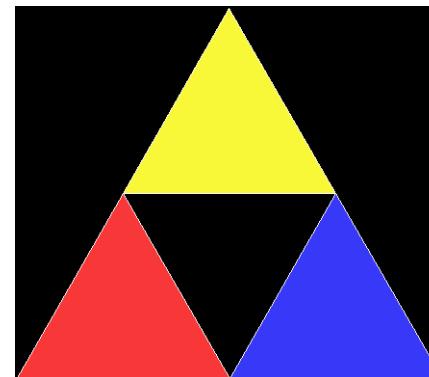
# Animated gifs: zooming on fractal



koch



Sierpinski



# Coordinate Transformations

Given a point  $p = \text{vector}(x_1, y_1, 1)$ , we can do:

## Translation

$$\begin{vmatrix} x_2 \\ y_2 \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ y_1 \\ 1 \end{vmatrix}$$

## Scaling

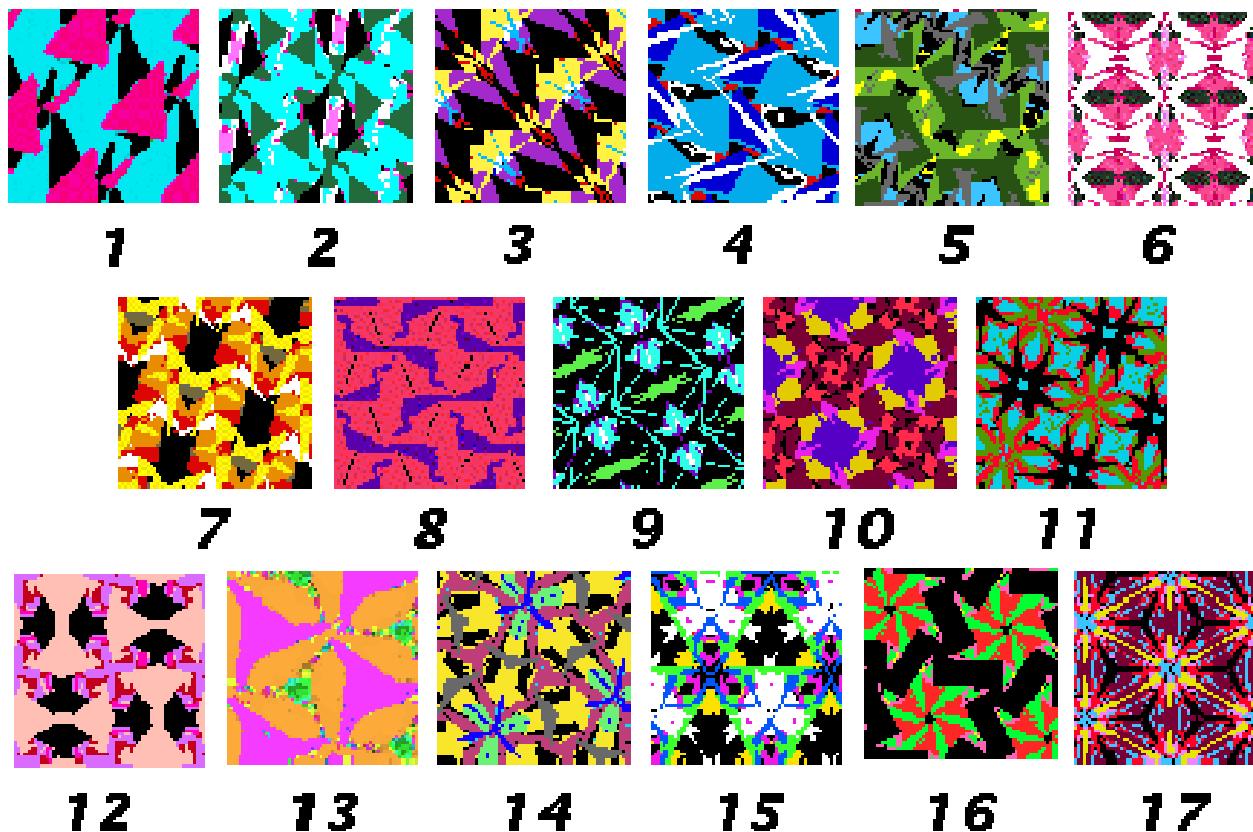
$$\begin{vmatrix} x_2 \\ y_2 \\ 1 \end{vmatrix} = \begin{vmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ y_1 \\ 1 \end{vmatrix}$$

## Rotation

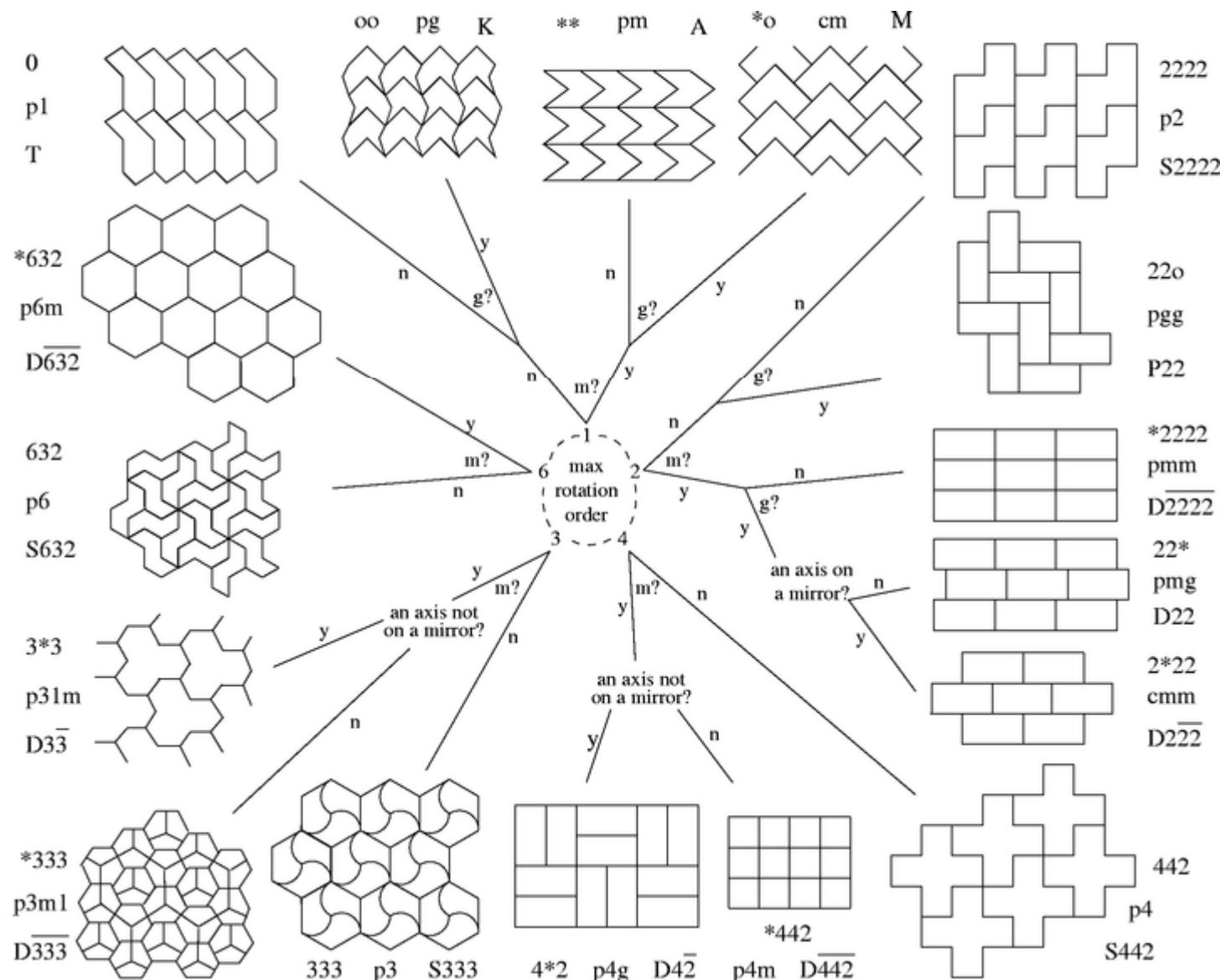
$$\begin{vmatrix} x_2 \\ y_2 \\ 1 \end{vmatrix} = \begin{vmatrix} \cos(v) & -\sin(v) & 0 \\ \sin(v) & \cos(v) & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ y_1 \\ 1 \end{vmatrix}$$

We can combine several matrix operations into a single matrix, because matrix multiplication is associative.

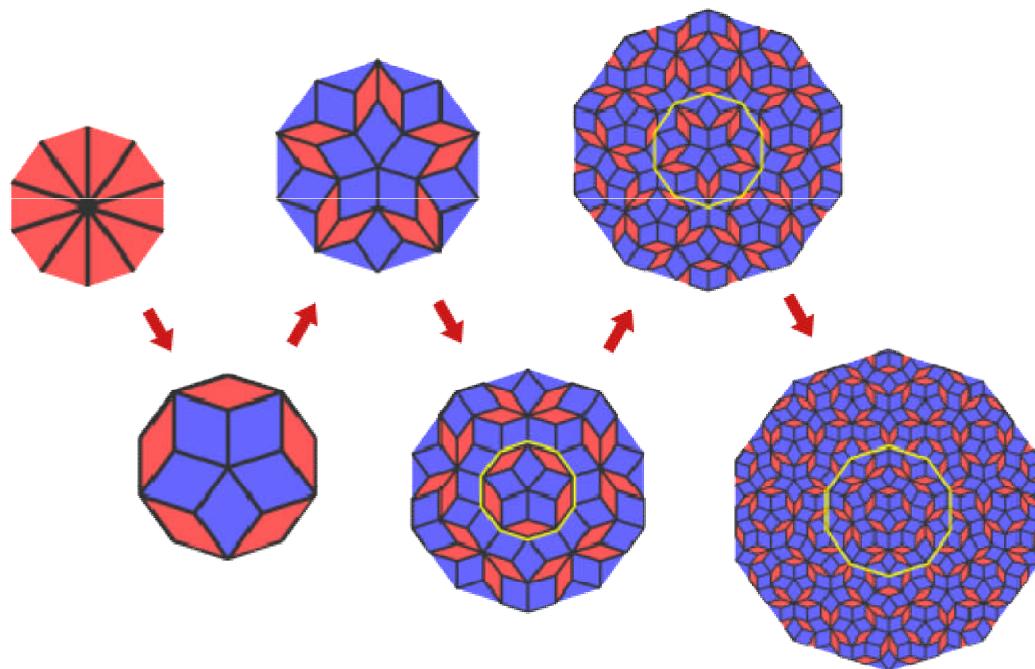
# Wallpaper patterns, only 17 different symmetric tilings are possible



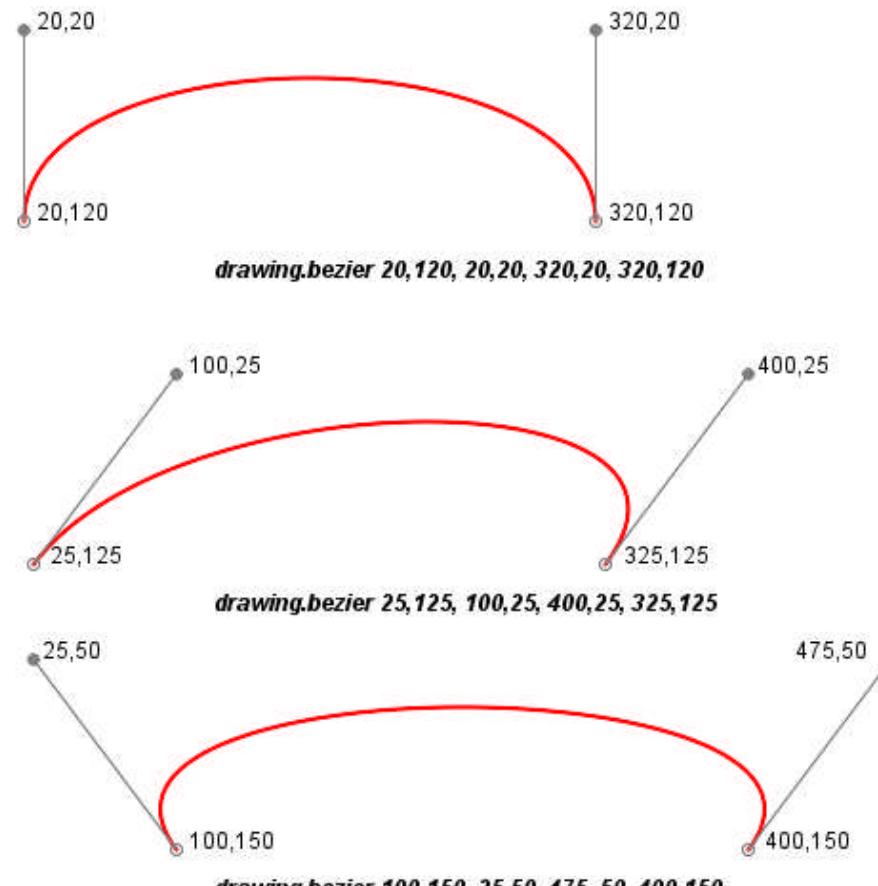
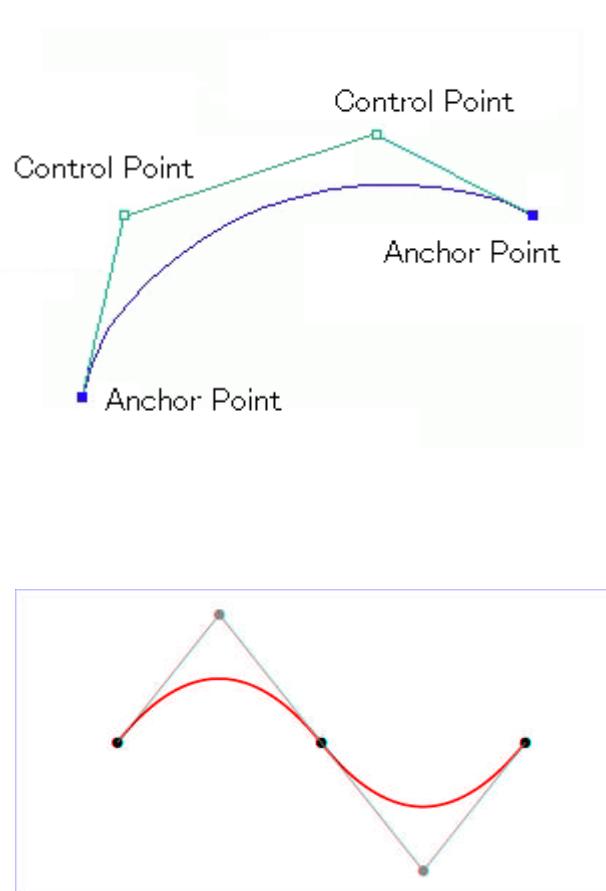
# The 17 Wallpaper groups



# Penrose non-periodic tilings



# Bézier curve

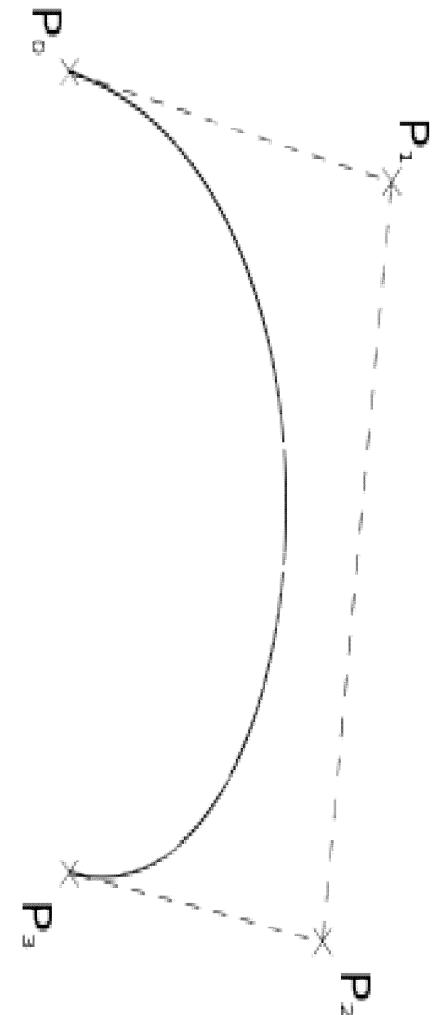


# Drawing a bezier curve

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

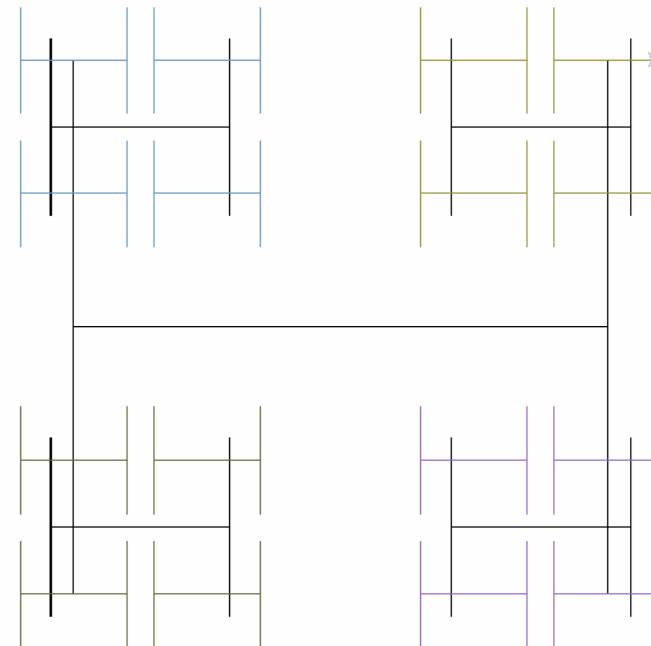
with  $t=[0..1]$ .

```
1. #!python3
2. import turtle
3. def turtle_cubic_bezier(x0, y0, x1, y1, x2, y2, x3, y3, n=20):
4. turtle.penup()
5. turtle.goto(x0,y0)
6. turtle.pendown()
7. for i in range(n+1):
8. t = i / n
9. a = (1. - t)**3
10. b = 3. * t * (1. - t)**2
11. c = 3.0 * t**2 * (1.0 - t)
12. d = t**3
13. x = int(a * x0 + b * x1 + c * x2 + d * x3)
14. y = int(a * y0 + b * y1 + c * y2 + d * y3)
15. turtle.goto(x,y)
16. turtle.goto(x3,y3)
17. turtle_cubic_bezier(0,0, 0,100, 100,0, 100,100)
```



# Homework

- Install python33
- Use turtle graphics to draw a figure recursively.
- Example, the H tree:



# References

## 1. L systems

<http://www.kevs3d.co.uk/dev/lsystems/>

# Python 2.X

# Reference

(or use google)

# Data types

- Numbers -- 3.1415, 1234, 999L, 3+4j
- Strings -- 'spam', "guido's"
- Lists -- [1, [2, 'three'], 4]
- Dict -- {'food':'spam', 'taste':'yum'}
- Tuples -- (1,'spam', 4, 'U')
- Files -- text = open('eggs', 'r').read()

# Constants

- 1234, -24, 0 -- integers (C longs)
- 999999999999L -- Long integers (unlimited size)
- 1.23, 3.14e-10, 4E210, 4.0e+210 -- Floats (C doubles)
- 0177, 0x9ff -- Octal and hex
- 3+4j, 3.0+4.0j, 3J -- Complex number

# Operators

- Operators -- Description
- lambda args: expression -- anonymous function
- x or y, -- Logical 'or' short-circuit
- x and y -- Logical 'and' short-circuit
- not x -- Logical negation
- in, not in -- sequence membership
- x | y -- Bitwise or
- x ^ y -- Bitwise exclusive or
- x & y -- Bitwise and
- x << y, x >> y -- Shift x left or right by y bits
- x + y, x - y -- Addition(concat, sub)
- x \* y, x / y, x % y -- Mult/repeat, div, mod/format
- -x, +x, ~x -- Unary negation, ident, bit complement
- x[i], x[i:j], x.y, x(...) -- Indexing, slicing, member, calls
- (...), [...], {...}, `...` -- Tuple, list, dictionary, to-string

# String operations

- `s1 = ""` -- Empty string
- `s2 = "spam's"` -- Double quotes
- `block = """..."""` -- Triple-quoted blocks
- `s1 + s2,` -- Concatenate,
- `s2 * 3` -- repeat
- `s2[i],` -- Index,
- `s2[i:j],` -- slice,
- `len(s2)` -- length
- `"a %s parrot" % 'dead'` -- String formatting
- `for x in s2,` -- Iteration,
- `'m' in s2` -- membership

# Slicing

```
>>> S = 'spam'
>>> S[0], S[-2] # indexing, front or end
('s', 'a')
>>> S[1:3], S[1:], S[:-1] # slice
('pa', 'pam', 'spa')
```

# String formatting

```
>>> exclamation = "Hi"
>>> "Who said %s!" % exclamation
'Who said Hi!'
>>> "%d %s %d you" % (1, 'like', 4)
'1 like 4 you'
>>> "%s -- %s -- %s" % (42, 3.14, [1, 2, 3])
'42 -- 3.14 -- [1, 2, 3]'
```

# Format specifiers

- % -- String (or any object's print format)
- %X -- Hex integer (uppercase)
- %c -- Character
- %e -- Floating-point format 1[6]
- %d -- Decimal (int)
- %E -- Floating-point format 2
- %i -- Integer
- %f -- Floating-point format 3
- %u -- Unsigned (int)
- %g -- Floating-point format 4
- %o -- Octal integer
- %G -- Floating-point format 5
- %x -- Hex integer
- %% -- Literal %

# Escaping chars

- \newline -- Ignored (a continuation)
- \n -- Newline (linefeed)
- \\ -- Backslash (keeps one \)
- \v -- Vertical tab
- \' -- Single quote (keeps ')
- \t -- Horizontal tab
- \" -- Double quote (keeps ")
- \r -- Carriage return
- \a -- Bell
- \f -- Formfeed
- \b -- Backspace
- \0XX -- Octal value XX
- \e -- Escape (usually)
- \xXX -- Hex value XX
- \000 -- Null (doesn't end string)
- \other -- Any other char (retained)

# List operations

- `L1 = []` -- An empty list
- `L2 = [0, 1, 2, 3]` -- Four items: indexes 0..3
- `L3 = ['abc', ['def', 'ghi']]` -- Nested sublists
- `L2[i], L3[i][j]` -- Index,
- `L2[i:j]`, -- slice,
- `len(L2)` -- length
- `L1 + L2,` -- Concatenate,
- `L2 * 3` -- repeat
- `for x in L2,` -- Iteration,
- `3 in L2` -- membership
- `L2.append(4),` -- Methods: grow,
- `L2.sort(),` -- sort,
- `L2.index(1),` -- search,
- `L2.reverse()` -- reverse, etc.
- `del L2[k],` -- Shrinking
- `L2[i:j] = []` -- slice assignment
- `L2[i] = 1,` -- Index assignment,
- `L2[i:j] = [4,5,6]` -- range(4), xrange(0, 4)

# Dictionary/Hash table operations

- `d1 = {}` -- Empty dictionary
- `d2 = {'spam': 2, 'eggs': 3}` -- Two-item dictionary
- `d3 = {'food': {'ham': 1, 'egg': 2}}` -- Nesting
- `d2['eggs'], d3['food']['ham']` -- Indexing by key
- `d2.has_key('eggs')`,  
test, -- Methods: membership
- `d2.keys()`, -- keys list,
- `d2.values()` -- values list, etc.
- `len(d1)`  
entries -- Length (number stored
- `d2[key] = new`, -- Adding/changing,
- `del d2[key]` -- deleting

# Tuple operations

- `()` -- An empty tuple
- `t1 = (0,)` -- A one-item tuple (not an expression)
- `t2 = (0, 1, 2, 3)` -- A four-item tuple
- `t2 = 0, 1, 2, 3` -- Another four-item tuple (same as prior line)
- `t3 = ('abc', ('def', 'ghi'))` -- Nested tuples
- `t1[i], t3[i][j]` -- Index, slice
- `t1[i:j], len(t1)` -- length
- `t1 + t2` -- Concatenate
- `t2 * 3` -- repeat
- `for x in t2,` -- membership
- `3 in t2` -- Iteration,

# File operations

- `output = open('/tmp/spam', 'w')` -- Create output file ('w' means write)
- `input = open('data', 'r')` -- Create input file ('r' means read)
- `S = input.read()` -- Read entire file into a single string
- `S = input.read(N)` -- Read N bytes (1 or more)
- `S = input.readline(marker)` -- Read next line (through end-line marker)
- `L = input.readlines()` -- Read entire file into list of line strings
- `output.write(S)` -- Write string S onto file
- `output.writelines(L)` -- Write all line strings in list L onto file
- `output.close()` -- Manual close (or it's done for you when collected)

# Mutable (can change)

- Object type -- Category--Mutable?
- Numbers -- Numeric--No
- Strings -- Sequence--No
- Lists -- Sequence--Yes
- Dictionaries -- Mapping--Yes
- Tuples -- Sequence--No
- Files -- Extension--N/A

# Booleans

- Object -- Value
- "text" -- True (any non empty string)
- "" -- False (empty string).
- [] -- False (empty list)
- {} -- False (empty set)
- 1 -- True
- 0.0 -- False (float number)
- None -- False

# Statements

- Statement -- Role -- Examples
- Assignment -- references -- curly, moe, larry = 'good', 'bad', 'ugly'
- Calls -- functions -- stdout.write("spam, ham, toast\n")
- Print -- Printing objects -- print 'The Killer', joke
- If/elif/else -- Selecting actions -- if "python" in text: print text
- For/else -- iteration -- for x in mylist: print x
- While/else -- General loops -- while 1: print 'hello'
- Pass -- placeholder -- while 1: pass
- Continue -- Loop jumps -- while 1: if not line: break
- Try/except/finally -- exceptions -- try: action() except: print 'error'
- Raise -- exception -- raise endSearch, location
- Import, From -- Module access -- import sys; from sys import stdin
- Def, Return -- functions -- def f(a, b, c=1, \*d): return a+b+c+d[0]
- Class -- Building objects -- class subclass: staticData = []
- Global -- Namespaces -- def function(): global x, y; x = 'new'
- Del -- Deleting things -- del data[k]; del data[i:j]; del obj.attr
- Exec -- Running code strings -- exec "import " + modName in gdict, Idict
- Assert -- Debugging checks -- assert X > Y

# keywords

and -- assert -- break -- class -- continue  
def -- del -- elif -- else -- except  
exec -- finally -- for -- from -- global  
if -- import -- in -- is -- lambda  
not -- or -- pass -- print -- raise  
return -- try -- while

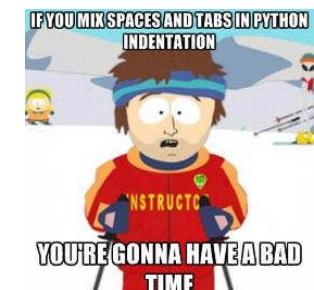
# Syntax

- Operation -- meaning
- cook(eggs, cheese) -- Function call
- eggs.cook( ) -- Method calls
- print eggs( ) -- Interactive print
- a < b and c != d -- Compound expression
- a < b < c -- Range tests

# Control: for, if, else

Printing prime numbers less than 10

```
>>> \n... for n in range(2, 10):\n... for x in range(2, n):\n... if n % x == 0:\n... print(n, 'factors', x, '*', n/x)\n... break # not a prime\n... else:\n... print(n, 'is a prime number')
```



# Advanced Python

- Lambdas
- Classes

# Main program

```
#!/usr/bin/python2.7
import os, string, sys, glob, operator, re

if __name__ == "__main__":
 print 'PYTHONPATH=',os.getenv("PYTHONPATH")
 sys.path.append('~/python')

if len(sys.argv) == 1:
 print >> sys.__stderr__, 'Usage:', \
sys.argv[0], '[options] args'
 sys.stderr.write('Usage: %s [options] args' % (sys.argv[0]))
 sys.exit()
for filename in sys.argv:
 print 'filename=',filename
```

# Parameter Passing

```
arg, *arg, **arg,kwargs
```

```

def pf(a,*b,**c):
 print "a=",a # a is object.
 print "b=",b # b is a tuple.
 print "c=",c # c is a dict.
```

---

pf(1,2,3)

a= 1

b= (2, 3)

c= { }

pf(1)

a= 1

b= ( )

c= { }

pf(1,2,3,4,5,e=1,f=1)

a= 1

b= (2, 3, 4, 5)

c= {'e': 1, 'f': 1}

# Regular vs Lambda functions (Anonymous functions)

```
> def sq(x): return x**2
```

```
> print sq(8)
```

64

```
> squarer = lambda x: x**2
```

```
> print squarer(8)
```

64

# Function Factory

```
def make_incremator(n): return lambda x: x + n
```

```
f = make_incremator(2);
g = make_incremator(6)
```

```
print f(42), g(42)
44 48
```

```
print make_incremator(22)(33)
55
```

# Filter, Map, Reduce

```
foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
```

```
print filter(lambda x: x % 3 == 0, foo)
[18, 9, 24, 12, 27]
```

```
print map(lambda x: x * 2 + 10, foo)
[14, 46, 28, 54, 44, 58, 26, 34, 64]
```

```
print reduce(lambda x, y: x + y, foo)
139
```

# Primes

```
>>> nums = range(2, 50) # nums is 2,3,4,5,..50
>>> for k in range(2, 8): # for k in 2,3,4,5,6,7,8
... nums = filter(lambda x: ((x == k) or (x % k)), nums)
... (Empty Line to end block of code)
>>> print nums
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

# Lambda string functions

```
sentence = 'It is raining cats and dogs'
```

```
words = sentence.split()
```

```
print words
```

```
['It', 'is', 'raining', 'cats', 'and', 'dogs']
```

```
lengths = map(lambda word: len(word), words)
```

```
print lengths
```

```
[2, 2, 7, 4, 3, 4]
```

```
print map(lambda w: len(w), 'It is raining cats and
dogs'.split())
```

```
[2, 2, 7, 4, 3, 4]
```

# List Comprehension

```
>>> vec = [2, 4, 6]
>>> [3*x for x in vec]
[6, 12, 18]
```

```
>>> [str(round(355/113.0, i)) for i in range(1,6)]
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

# Dict, Hash Maps

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
```

# Record / Structs

```
class Record: pass # empty class
x = Record() # constructor.
x.name='John'
x.address='USA'
```

# OO-Classes

```
class FirstClass: # base class object
 def setdata(self, value): # ctor,cons
 self.data = value # self is this instance
 def display(self):
 print self.data # self.data: per instance
=====
class SecondClass(FirstClass): # inherits data
 def display(self): # overrides base display
 print 'Current value = "%s"' % self.data
=====
class ThirdClass(SecondClass): # is-a SecondClass
 def __init__(self, value):
 self.data = value
 def __add__(self, other): # on "self + other"
 return ThirdClass(self.data + other) # New copy.
 def __mul__(self, other):
 self.data = self.data * other
```

# OO-Classes

```
a=ThirdClass("abc")
a.display()
Current value = "abc"
=====
b = a + 'xyz'
ThirdClass.__add__ called: makes a new instance
b.display()
Current value = "abcxyz"
=====
a * 3
ThirdClass.__mul__ called: changes instance in-
place
a.display()
Current value = "abcabcabc"
```