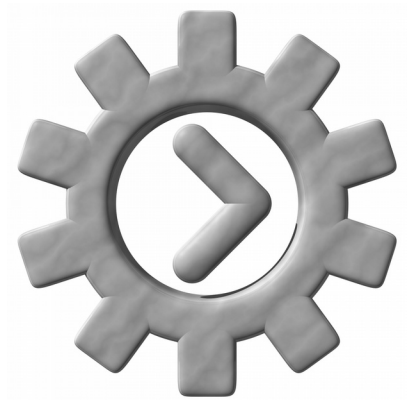


# **100 Command Line Tools**

## **For Windows, Linux and Mac OS/X**



**How to do things fast,  
with the same commands,  
on every computer!**

Informations in this book are provided by the author "as is" and any express or implied warranties, including, but not limited to, the implied warranties of fitness for a particular purpose are disclaimed. In no event shall the publisher or author be liable for any direct, indirect, incidental, special, exemplary or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of the software described here, even if advised of the possibility of such damage.

---

## Contents

### Introduction

How to do things the same way on all computers	10
How to get the SFK tool running instantly	11
The different editions of SFK: OSE, Base, XE	14

### First steps: the SFK tutorial

#### File handling

List all files of a folder, and all sub folders	15
List only selected files in selected sub folders	16
List files using wildcards	17
List the latest or biggest files	17
Find a filename quickly in the current directory tree	18
List different files between two folders	19
Run a command on all files of a folder	19
Rename files quickly using patterns	21
List the size of directory tree contents	23
Copy a folder, or parts of it, or only updates	24
Delete or clean up specific files in a folder	26
How to use index files for fast filename lookup	27
Tell where in the PATH a command is run from	29
Create checksums of files	30
Find duplicate files	31

#### Find and replace within files

Find words in text and binary files	32
Replace words in text and binary files	33
Flexible filter and replace in a single text file	34
Search in files using wildcards and Simple Expressions	35

---

**File conversion and processing**

Convert plain text files between Windows/Linux format	36
Remove TAB characters from text	37
Split large files	38
Collect many text files into one large text	39
Sort text lines alphabetically	40

**Send files via network**

How to send a file from one computer to another	42
How to transfer many files, or just changed ones	44

**Further useful functions**

Read or write the clipboard under Windows	46
Convert CSV data to tab separated text	48
Count text lines	41
Write long commands into a script	50
Search environment variables for words	52

**Xed big examples**

reformat comma separated value (CSV) text	54
convert fixed column data to CSV data	55
convert CSV data to XML data	56
convert XML data to CSV data	58
cleaning up a translation file	60
extract 2 letter phrases from text	61
Wiki markup text to HTML code conversion	62
HTTP Scripting and Test Automation	67
Filling an XML file with program meta informations	72
A detailed +perline example	79
Script creation and debugging tips	82

---

---

## SFK Command Reference

general infos	85	syntax concepts valid for most commands
Windows/Linux/Mac syntax differences	87	what needs to be typed different in the examples under Linux and Mac OS/X
<b>file system</b>		
sfk list	88	list directory contents
sfk filefind	93	find files by filename
sfk treesize	94	show directory size statistics
sfk copy	96	copy directory trees additively
sfk sync	96	mirror tree content with deletion
sfk rename	99	flexible multi file rename
sfk partcopy	126	copy part from a file into another one
sfk mkdir	133	create directory tree
sfk delete	133	delete files and folders
sfk deltree	133	delete whole directory tree
sfk deblank	135	remove blanks in filenames
sfk space [-h]	135	tell total and free size of volume
sfk filetime	136	tell times of a file
sfk touch	138	change times of a file
sfk index	140	create index file(s) for fast lookup
sfk name	142	lookup file names using index files
sfk fixfile	305	Windows: change bad filenames and times
sfk setbytes	128	set bytes at offset within a file compression
<b>compression</b>		
sfk zip	106	create zip file from folder
sfk zipto	109	zip selected file list
sfk unzip	111	list or extract zip file
sfk checkzip	113	verify zip file content

---

**conversion**

<code>sfk lf-to-crlf</code>	145	convert from LF to CRLF line endings
<code>sfk crlf-to-lf</code>	146	convert from CRLF to LF line endings
<code>sfk oload</code>	360	load office file content as text
<code>sfk detab</code>	148	convert TAB characters to spaces
<code>sfk entab</code>	149	convert groups of spaces to TAB chars
<code>sfk scantab</code>	150	list files containing TAB characters
<code>sfk split</code>	151	split large files into smaller ones
<code>sfk join</code>	152	join small files into a large one
<code>sfk csvtotab</code>	153	convert .csv data to tab separated
<code>sfk tabtocsv</code>	154	convert tab separated to .csv format
<code>sfk encode</code>	114	convert data to base64 or hex format
<code>sfk decode</code>	114	decode base64, hex or url format
<code>sfk wtoa</code>	115	convert wide chars to Ansi
<code>sfk wtou</code>	116	convert wide chars to UTF-8
<code>sfk utoa</code>	119	convert UTF-8 text to Ansi
<code>sfk hexdump</code>	155	create hexdump from a binary file
<code>sfk hextobin</code>	158	convert hex data to binary
<code>sfk hex</code>	159	convert decimal number(s) to hex
<code>sfk dec</code>	160	convert hex number(s) to decimal
<code>sfk chars</code>	160	print chars for a list of codes
<code>sfk bin-to-src</code>	162	convert binary to source code

**text processing**

<code>sfk filter</code>	163	search, filter and replace text data
<code>sfk ofilter</code>	163	filter text from an office file
<code>sfk replace</code>	173	replace words in binary and text files
<code>sfk xed</code>	187	edit stream text using sfk expressions
<code>sfk xex</code>	181	extract text from stream using expressions
<code>sfk run</code>	201	run command on all files of a folder
<code>sfk runloop</code>	205	run a command n times in a loop
<code>sfk printloop</code>	205	print some text many times
<code>sfk load</code>	359	load file content for further processing
<code>sfk perline</code>	206	run sfk command(s) per input text line
<code>sfk head</code>	209	print first lines of a file
<code>sfk tail</code>	210	print last lines of a file
<code>sfk snapto</code>	214	join many text files into one file
<code>sfk addhead</code>	215	insert string at start of text lines

---

sfk addtail	216	append string at end of text lines
sfk joinlines	150	join text lines split by email reformatting
sfk strings	216	extract strings from a binary file
sfk sort	217	sort text lines produced by another command
sfk count	218	count text lines, filter identical lines
sfk lineLen	219	tell length of string(s)
<b>search and compare</b>		
sfk xfind	227	search in text files using wildcards and simple expressions
sfk ofind	239	search in office files .docx .xlsx .ods
sfk xfindbin	227	search in text and binary files
sfk xhexfind	240	search with hexdump output
sfk extract	244	extract data from text and binary
sfk find	221	search static text, without wildcards
sfk hexfind	224	search static binary data
sfk md5gento	249	create list of md5 checksums over files
sfk md5check	250	verify list of md5 checksums over files
sfk md5	251	calc md5 over a file, compare two files
sfk pathfind	253	search PATH for location of a command
sfk refflist	254	list fuzzy references between files
sfk deplist	256	list fuzzy dependencies between files
sfk dupfind	258	find duplicate files by content
<b>networking</b>		
sfk httpserv	260	run an instant HTTP server. type "sfk httpserv -help" for help.
sfk ftpserv	261	run an instant FTP server type "sfk ftpserv -help" for help.
sfk ftp	265	instant anonymous FTP client
sfk web	273	send HTTP request to a server
sfk wget	270	download HTTP file from the web
sfk tcpdump	275	print TCP conversation between programs
sfk udpdump	277	print incoming UDP requests
sfk udpsend	279	send UDP requests
sfk ip	284	tell own machine's IP address(es). type "sfk ip -help" for help.
sfk netlog	280	send text outputs to network, and/or file, and/or terminal

---

---

sfk ping	129	ping multiple machines in one go
<b>scripting</b>		
sfk help chain	356	how to combine multiple commands
sfk script	286	run many sfk commands in a script file
sfk label	289	define starting points within a script
sfk call	290	call a sub function at a label
sfk echo	302	print (coloured) text to terminal
sfk color	305	change text color of terminal
sfk setvar	366	put text into an sfk variable
sfk storetext	293	store text in memory for later use
sfk alias	306	create command from other commands
sfk mkcd	308	create command to reenter directory
sfk sleep	310	delay execution for milliseconds
sfk pause	310	wait for user input
sfk stop	293	stop sfk script execution
sfk tee	311	split command output in two streams
sfk tofile	311	save command output to a file
sfk toterm	311	flush command output to terminal
sfk for	208	repeat commands many times
sfk loop	312	repeat execution of a command chain
sfk cd	312	change directory within a script
sfk getcwd	313	print the current working directory
sfk require	313	compare version text
sfk time [-h]	314	print current date and time
<b>development</b>		
sfk bin-to-src	162	convert binary data to source code
sfk make-random-file	315	create file with random data
sfk fuzz	316	change file at random, for testing
sfk sample	317	print example code for programming
sfk patch	220	change text files through a script
sfk inst	323	instrument c++ with tracing calls
<b>diverse</b>		
sfk status	131	send colored status to the SFKTray Windows GUI utility for display
sfk calc	294	do a simple instant calculation
sfk random	105	create a random number

---



sfk prompt	300	ask for user input
sfk number	301	print number in diverse formats
sfk xmlform	324	reformat xml for easy viewing
sfk media	325	cut video and binary files
sfk view	328	show results in a GUI tool
sfk toclip	332	copy command output to clipboard
sfk fromclip	332	read text from clipboard
sfk env	333	search environment variables
sfk version	334	show version of a binary file
sfk ascii	337	list ISO 8859-1 ASCII characters
sfk ascii -dos	337	list OEM codepage 850 characters
sfk spell [-h]	335	phonetic spelling for telephone
sfk cmd	336	print an example command
sfk data	212	create random test data
sfk ruler	208	measure console text width

### help by subject

sfk samp	317	example scripts on sfk use
sfk help select	338	how dirs and files are selected in sfk
sfk help office	361	how to search in office files
sfk help options	344	general options reference
sfk help patterns	350	wildcards and text patterns within sfk
sfk help chain	356	how to combine (chain) multiple commands
sfk help var	363	how to use sfk variables and parameters
sfk help shell	371	how to optimize the windows cmd.exe
sfk help chars	352	about locale specific characters
sfk help nocase	354	about case insensitive search
sfk help unicode	369	about wide character file reading support
sfk help colors	373	how to change result colors
sfk help compile	370	how to compile sfk on any linux system
sfk help xe	375	for infos on sfk extended edition.

### commercial commands available only with SFK XE

sfk xreplace	193	replace in files using sfk expressions
--------------	-----	--

## Index

alphabetical index	377
--------------------	-----

## Introduction

### How to do things the same way on all computers

Quick file exchange between machines, find duplicate files, find and replace text, list directory tree sizes, and many other functions for daily tasks: normally this requires hour-long installations and configurations of masses of separate programs, often interrupted by missing or wrong versioned libraries ("do you have the latest .NET/Java/Cygwin/Qt?"). There may be a thousand tools for the same task, but you may have to install 5 of them, just to find out they work different than expected, until the 6th may work, causing a spammed registry and time waste in general.

Therefore the **Swiss File Knife** - aka **SFK** - was developed. It is a small, single binary for the command line that runs instantly, without installation. It contains 100 tools for the most needed tasks, with the same basic syntax for Windows, Linux and Mac OS/X.

---

## How to get the Swiss File Knife up and running anywhere.

### Download the executables for Windows, Linux or Mac OS/X

By web browser: go to

<http://stahlworks.com/sfk/>

then click on one of the top links to download your binary instantly.

Alternatively, look on SourceForge:

<http://sourceforge.net/projects/swissfileknife/>

or on a Linux text console, use one of these:

```
wget http://stahlworks.com/sfkux      for current 32 bit systems
wget http://stahlworks.com/sfkux64   for current 64 bit systems
wget http://stahlworks.com/sfkuxold  for older 32 bit systems
                                       (like DSL, using lib5)
wget http://stahlworks.com/sfkarm    for 32 bit ARM systems
```

If your system has no wget command then try curl instead, like:

```
curl -o sfk http://stahlworks.com/sfkux
```

The Apple Mac OS/X binaries are available by:

```
curl -o sfk http://stahlworks.com/sfkmac      for current Intel
                                                based Macs
curl -o sfk http://stahlworks.com/sfkmacold   for PowerPC based
                                                Macs
```

Self compile: on systems for which no binary is available you may download the sourcecode from the SourceForge link (.zip or .tar.gz). Make sure the g++ or gcc compiler is installed on your system. Then type:

```
g++ sfk.cpp sfkext.cpp sfkpack.cpp -o sfk
```

### Transfer of SFK without internet access:

If the target machine has any connection to a local network, try the following:

#### SFK Instant HTTP Server for easy file exchange

on another machine where you have SFK already, type

```
sfk httpserv -port=9090
```

and make sure the sfk binary is located in the current folder.

on the target machine, open a web browser and access:

```
http://othermachine:9090/
```

---

or on a Linux/Mac console, type one of:

```
wget http://othermachine:9090/sfk
curl -o sfk http://othermachine:9090/sfk
```

*further reading:*

httpserv tutorial on page 42,  
reference on page 260.

If that fails (no browser, no gui, no wget or curl command), check if there is an "ftp" command on the target. If so, try:

### **SFK Instant FTP Server for easy file exchange**

on a machine where you have SFK already, type:

```
sfk ftpserv
```

it will tell you the machine's IP address. then, on the target machine, type:

```
ftp ipaddress
```

and if the login succeeds, try:

```
bin
get sfk.exe
```

If ftp cannot connect to the server then try to run ftpserv as administrator. If get fails, check if the ftp client on the target accepts the command:

```
passive
```

then try to "get" again (ftp creates a new connection per file download, which is often blocked by firewalls. the passive command changes the way in which those connections are created.)

*further reading:*

ftpserv tutorial on page 42,  
or the reference on page 261.

### **How to prepare the SFK binary under Linux:**

After download, you have to type

```
mv sfkux sfk
chmod +x sfk
```

to enable execution (the 'x' flag) of sfk. Then simply type

```
./sfk
```

to get it running (the "./" is often needed as the PATH may not contain the current directory ".").

---

## Where to place the SFK executable:

### Recommendation for Windows

Create a directory structure

```
c:\app\bin
```

then copy sfk.exe to c:\app\bin. Then extend the Windows Shell Path like

```
set PATH=%PATH%;c:\app\bin
```

which is best done in a batch file like c:\app\init.bat, so after opening CMD.EXE just type c:\app\init to extend the path. Also make sure your Windows Shell (CMD.EXE) supports command auto-completion and copy/paste of text (the QuickEdit and Insert setting), otherwise it is very hard to use!

*further reading:  
Windows CMD.EXE configuration,  
sfk help shell on page 371.*

If you create a collection of batch files (e.g. through the "sfk alias" command on page 306) it is most convenient to store them in c:\app\bin as well, as this path is short and contains no blank characters. Further tools can be installed parallel to "bin" into c:\app.

### Recommendation for Linux and Mac OS/X

Type "cd" then "pwd" to find out what your account's home directory is. Within your home directory (e.g. /home/users/youruserid/) create a directory "tools" by

```
mkdir tools
```

then rename sfk-linux.exe to sfk, and copy that into the tools dir.

Extend the PATH like:

```
export PATH=$PATH:/home/users/youruserid/tools
```

then you should be able to run sfk by typing "sfk".

By default, there are no colors, as it is not possible to autodetect the background color under Linux/Mac. If you like colorful output then read on under "sfk help color" on page 373.

---

## The different editions of SFK: OSE, Base, XE

Three different types of SFK binaries exist:

- **SFK OSE** - the Open Source Edition: this is what you get when compiling the available source codes. It contains 90 percent of SFK functionality but not the commands `xfind`, `xhexfind`, `xex`, `extract` as well as the commercial commands `xed` and `xreplace`.
- **SFK Base+XD**: these are the binaries you can download from [stahlworks.com](http://stahlworks.com) and SourceForge. They also contain the closed source functions `xfind`, `xhexfind`, `xex` and `extract`, as well as the help text for `xed`, and a demo of `xreplace` that cannot write files.
- **SFK XE or Extended Edition**: this is the commercial edition of SFK available from [stahlworks.com](http://stahlworks.com). It contains a high performance implementation of `sfk replace`, the `sfk xed` command and the `sfk xreplace` command. Furthermore it can search `.zip`, `.tar.gz` and `.tar.bz2` file contents directly with several commands. SFK Base+XD contains a demo of this, allowing to read the first 1000 bytes of every `.zip` file entry.

Some Linux distributions allow installation of SFK via their package managers, however these will be SFK OSE binaries. If in any doubt, type

```
sfk ver -own
```

and it will tell it's own edition, like:

```
Base/XD windows-any 1.9.3
```

---

## SFK Tutorial

A step by step introduction into the most popular commands of Swiss File Knife.

### List all files of a folder, and all sub folders

Everyone knows that `dir mydir` on Windows, or `ls mydir` on Linux/Mac shows the filenames in the top level of a folder `mydir`, without it's sub folders.

If, however, you want to list all files in `mydir` and all it's sub folders, as a flat list of filenames with full path each, then use

```
sfk dir mydir
```

example output:

```
mydir\project1\01-make-all.sh
mydir\project1\app\gui\base\Tools.cpp
mydir\project1\app\gui\base\Tools.hpp
mydir\project1\app\gui\login\Screen.cpp
mydir\project1\config.h
mydir\project1\config.h.bak
mydir\project1\save\config.h
mydir\project1\save2\config.h
mydir\project1\save3\config.h
mydir\project1\tmp\trash1.txt
mydir\project1\tmp\trash2.txt
mydir\project1\tmp\trash3.txt
mydir\project1\tools\include\Tools.hpp
mydir\project1\tools\include\Tools.hpp.bak
mydir\project1\tools\new.myscm\sub1.txt
mydir\project1\tools\org.myscm\sub1.txt
mydir\project1\tools\source\myscm\sub3.txt
mydir\project1\tools\source\other1.myscm
mydir\project1\tools\source\other1.myscm.bak
mydir\project1\tools\source\save\myscm
mydir\project1\tools\source\save\myscm-file.txt
mydir\project1\tools\source\save\Tools.cpp
mydir\project1\tools\source\Tools.cpp
mydir\project1\tools\source\Tools.tmp
25 files, 18 dirs, 2828 bytes.
```

Notice that sub folder traveling is **default** with most SFK commands, so you don't have to use an extra option for that. This is because, if I want to do something "with all files of a folder", in most cases I literally mean **all** files.

Instead of "sfk dir" you may also use "sfk list" which produces just the list of filenames, without the "files, dirs, bytes" info.

## List only selected files in selected sub folders

In the above example, we notice two kinds of files:

- live files we are actively working with
- backup or trash files and folders named tmp, bak, save.

In most cases, we want to

- list all files of that folder
- except for files within folders having tmp or save in their name
- and except for files ending with .bak or .tmp.

This can be done with SFK by:

```
sfk dir -dir mydir -subdir !tmp !save -file !.bak !.tmp
```

example output:

```
mydir\project1\01-make-all.sh
mydir\project1\app\gui\base\Tools.cpp
mydir\project1\app\gui\base\Tools.hpp
mydir\project1\app\gui\login\Screen.cpp
mydir\project1\config.h
mydir\project1\tools\include\myscm\sub2.txt
mydir\project1\tools\include\Tools.hpp
mydir\project1\tools\new.myscm\sub1.txt
mydir\project1\tools\org.myscm\sub1.txt
mydir\project1\tools\source\myscm\sub3.txt
mydir\project1\tools\source\other1.myscm
mydir\project1\tools\source\Tools.cpp
12 files, 13 dirs, 1376 bytes.
```

**Wildcards are default** and need not to be specified in most cases. This means that !save actually means !\*save\* - i.e. excluding every sub directory that has save somewhere in it's name, like save, save2, save3 etc.

**Under Linux/Mac** you have to use a colon ":" instead of "!" because the command shell misinterprets "!" as some command for itself.

So use instead:

```
sfk dir -dir mydir -subdir :tmp :save -file :.bak :.tmp
```



## Listing files using wildcards

To list files within sub folder names containing words "new" and "scm" use

```
sfk list -dir mydir -subdir new*scm
```

example output:

```
mydir\project1\tools\new.myscm\sub1.txt
```

**Under Linux/Mac** you must surround anything with \* or ? by double quotes because the command shell misinterprets "\*" as some command for itself.

Alternatively you may use % as a replacement for "\*". So use one of:

```
sfk list -dir mydir -subdir "new*scm"  
sfk list -dir mydir -subdir new%scm
```

*for all Linux/Mac syntax  
details see page 87.*

## List the latest or biggest files

Which files were changed most recently within mydir? Find out by:

```
sfk list -late mydir
```

example output:

```
2015-01-18 06:47:54 mydir\project1\app\gui\base\Tools.cpp  
2015-01-18 13:44:17 mydir\project1\tools\source\save\myscm  
2015-02-28 08:54:20 mydir\project1\tools\source\other1.myscm  
2015-02-28 08:54:20 mydir\project1\tools\source\Tools.cpp  
2015-02-28 08:54:20 mydir\project1\tools\source\Tools.tmp
```

And what are the biggest files in mydir?

```
sfk list -big mydir
```

example output:

```
41 mydir\project1\save2\config.h  
56 mydir\project1\save\config.h  
171 mydir\project1\config.h  
1074 mydir\project1\tools\source\Tools.cpp  
1210 mydir\project1\tools\source\Tools.tmp
```

*further reading:  
sfk list, the full syntax  
reference on page 88.*

## Find a filename quickly in the current directory tree

You are standing within a folder and know that a file having foo somewhere in its path- and/or filename exists. But you don't know exactly where. This can be solved by

```
sfk filefind foo
```

example output:

```
project1\tools\source\BarFoo.cpp
```

So, there is a file "BarFoo" in a sub folder project1\tools\source . Notice that **case insensitive search is default** with every SFK command, therefore "foo" finds both "foo" and "Foo". Because this quick local filename search is needed so often, you may also type:

```
sfk :foo
```

Which does the same as "filefind foo".

Another example:

```
sfk :tool*sub2
```

may find:

```
project1\tools\include\myscm\sub2.txt
```

as this contains "tool" in its **path** and "sub2" in its **filename**.

**Under Linux/Mac** use instead:

```
sfk :tool%sub2
```

as otherwise a \* wildcard would be misinterpreted by the shell and not given to SFK.

*full syntax and examples  
in the reference on page 93.*

## List different files between two folders

I have files in a folder "step1". I make a copy of the whole folder as "step2" and continue working within "step2". After some hours I wonder which files are different compared to the old folder.

```
sfk list -sincedir step1 step2
```

tells:

```
[dif] step2\base.php
[dif] step2\classes\tree.class.php
[dif] step2\index.php
[add] step2\organizer.php
[add] step2\tasks.php
```

meaning:

- 3 files that exist in both folders are different
- 2 files have been created in step2 that did not exist in step1

Note that files which were deleted in folder step2 are not shown. These can be found by running a reverse folder comparison:

```
sfk list -sinceadd step2 step1
```

tells:

```
[add] step1\queuescanner.php
```

so the file queuescanner.php was deleted in step2.

*find all sfk list options  
on page 88.*

## Run a command on all files of a folder

I want to collect all .jpg files in a folder mydir like

```
mydir\Formats\06-binary.jpg
mydir\myproj\app\gui\base\GreenFoo.jpg
mydir\myproj\app\gui\login\Door.jpg
mydir\myproj\tools\BackButton.jpg
mydir\myproj\tools\Home.jpg
```

into a single flat folder called "overview". This can be done by:

```
sfk list mydir .jpg +run "copy %qfile overview"
```

on Linux/Mac: use #qfile instead of \$qfile.

example output:

```
[simulating:]
copy "mydir\Formats\06-binary.jpg" overview
```

```
copy "mydir\myproj\app\gui\base\GreenFoo.jpg" overview
copy "mydir\myproj\app\gui\login\Door.jpg" overview
copy "mydir\myproj\tools\BackButton.jpg" overview
copy "mydir\myproj\tools\Home.jpg" overview
[add -yes to execute.]
```

remarks:

- first list the files by "sfk list".
- then add "+run" as a chained command.
- qfile means "filename enclosed in double quotes",  
in case that any filename contains blank characters.
- finally add "-yes" to really run the commands.

*full sfk run syntax  
on page 201.*

## Run a command on files that differ between two folders

Back to the example where I listed all files different between two folders named "step1" and "step2". If I want to run a difference viewer tool like WinMerge on the different files this can be done by:

```
sfk list -sincendiff step1 step2 +run "winmerge $qsince $qfile"
```

on Linux/Mac: use # instead of \$.

remarks:

- now we don't use -sincendir but -sincendiff so it will select only *files that exist in both folders and which are different*, but not *added* files.
- then this is passed to +run
- \$qsince references the filename from step1, \$qfile the one from step2,  
and again "q" means to enclose all filenames in double quotes
- this assumes that a tool winmerge.exe exists in a folder listed in the PATH environment variable .

the simulation output tells exactly what would be done:

```
[simulating:]
winmerge "step1\base.php" "step2\base.php"
winmerge "step1\classes\tree.class.php"
        "step2\classes\tree.class.php"
winmerge "step1\index.php" "step2\index.php"
[add -yes to execute.]
```

finally add -yes to run the commands. To stop inbetween press CTRL+C.

*full sfk run syntax  
on page 201.*