

# **The Swiss File Knife Book**

**Version 1.9.3**



**Copyright (c) 2018 by StahlWorks Technologies**  
**[www.stahlworks.com](http://www.stahlworks.com)**

**Only for personal use on devices of the PDF file purchaser.**  
**Printing is allowed for up to three copies per purchased PDF file.**

## Table of contents

Touch any number to jump to that page.  
In every page, touch on contents to jump back here.

### Introduction

How to do things the same way on all computers	8	
How to get the SFK tool running instantly		9
The different editions of SFK: OSE, Base, XE		12

### Tutorial 13

#### File handling

List all files of a folder, and all sub folders	13	
List only selected files in selected sub folders		14
List files using wildcards	15	
List the latest or biggest files	15	
Find a file quickly in the current directory tree		16
List different files between two folders	17	
Run a command on all files of a folder		18
Rename files quickly using patterns		20
List the size of directory tree contents	22	
Copy a folder, or parts of it, or only updates		23
Delete or clean up specific files in a folder		25
How to use index files for fast filename lookup	26	
Tell where in the PATH a command is run from		28
Create checksums of files		29
Find duplicate files	30	

#### Find and replace within files

Find words in text and binary files	31	
Replace words in text and binary files		32
Flexible filter and replace in a single text file	33	
Search in files using wildcards and Expressions		34

**File conversion and processing**

Convert plain text files between Windows/Linux format	35
Remove TAB characters from text	36
Split large files	37
Collect many text files into one large text	38
Sort text lines alphabetically	39

**Send files via network**

How to send a file from one computer to another	41
How to transfer many files, or just changed ones	43

**Further useful functions**

Read or write the clipboard under Windows	45
Convert CSV data to tab separated text	47
Count text lines	40
Write long commands into a script	49
Search environment variables for words	51

**Xed detailed examples** 52

reformat comma separated data	53
convert fixed column data to CSV	54
convert CSV data to XML data	55
convert XML data to CSV data	57
cleaning up a translation file	59
extract two letter words from text	60

**Large script examples**

Wiki markup text to HTML conversion using sfk script and xed	61
HTTP Scripting and Test Automation using sfk script, call, web, perline, xex	66
Filling an XML file with program meta data using sfk script, xex, version, list, calc, filter, setvar, getvar, xed	71

**A detailed perline example** 78**Script creation and debugging tips** 81

## Command Reference

General infos		83
Windows/Linux/Mac syntax differences		85
<b>file system</b>		
<b>sfk list</b>	- list directory tree contents. list latest, oldest or biggest files. list directory differences. list zip jar tar gz bz2 contents.	86
<b>sfk filefind</b>	- find files by filename	243
<b>sfk treesize</b>	- show directory size statistics	107
<b>sfk copy</b>	- copy directory trees additively	205
<b>sfk sync</b>	- mirror tree content with deletion	205
<b>sfk rename</b>	- flexible multi file rename	208
<b>sfk partcopy</b>	- copy part from a file into another one	191
<b>sfk mkdir</b>	- create directory tree	214
<b>sfk delete</b>	- delete files and folders	104
<b>sfk deltree</b>	- delete whole directory tree	104
<b>sfk deblank</b>	- remove blanks in filenames	155
<b>sfk space [-h]</b>	- tell total and free size of volume	201
<b>sfk filetime</b>	- tell times of a file	203
<b>sfk touch</b>	- change times of a file	235
<b>sfk index</b>	- create index file(s) for fast lookup	91
<b>sfk name</b>	- lookup file names using index files	93
<b>sfk fixfile</b>	- change bad filenames and file times	268
<b>sfk setbytes</b>	- set bytes at offset within a file	321
<b>compression</b>		
<b>sfk zip</b>	- create zip file from folder	325
<b>sfk zipto</b>	- zip selected file list	328
<b>sfk unzip</b>	- list or extract zip file	330
<b>sfk checkzip</b>	- verify zip file content	333
<b>conversion</b>		
<b>sfk lf-to-crlf</b>	- convert from LF to CRLF line endings	105
<b>sfk crlf-to-lf</b>	- convert from CRLF to LF line endings	107
<b>sfk detab</b>	- convert TAB characters to spaces	102
<b>sfk entab</b>	- convert groups of spaces to TAB chars	103
<b>sfk scantab</b>	- list files containing TAB characters	103

<b>sfk split</b>	- split large files into smaller ones	189	
<b>sfk join</b>	- join small files into a large one		190
<b>sfk csvtotab</b>	- convert .csv data to tab separated	233	
<b>sfk tabtocsv</b>	- convert tab separated to .csv format		234
<b>sfk encode</b>	- convert data to base64 or hex format		315
<b>sfk decode</b>	- decode base64, hex or url format	315	
<b>sfk wtoa</b>	- Windows: convert wide chars to Ansi		316
<b>sfk wtou</b>	- convert wide chars to UTF-8	318	
<b>sfk utoa</b>	- convert UTF-8 text to Ansi		319
<b>sfk hexdump</b>	- create hexdump from a binary file		177
<b>sfk hextobin</b>	- convert hex data to binary	247	
<b>sfk hex</b>	- convert decimal number(s) to hex		260
<b>sfk dec</b>	- convert hex number(s) to decimal		260
<b>sfk chars</b>	- print chars for a list of codes	246	
<b>sfk bin-to-src</b>	- convert binary to source code		134

text processing

<b>sfk filter</b>	- search, filter and replace text data	135	
<b>sfk xed</b>	- edit stream text using expressions		225
<b>sfk xex</b>	- extract from stream text using expressions		229
<b>sfk addhead</b>	- insert string at start of text lines	145	
<b>sfk addtail</b>	- append string at end of text lines		145
<b>sfk patch</b>	- change text files through a script		146
<b>sfk snapto</b>	- join many text files into one file	100	
<b>sfk joinlines</b>	- join text lines split by email client		101
<b>sfk inst</b>	- instrument c++ sourcecode with tracing calls		153
<b>sfk replace</b>	- replace words in binary and text files	217	
<b>sfk xreplace</b>	- replace in files using expressions	127	
<b>sfk hexfind</b>	- find words in binary files, showing hexdump		111
<b>sfk run</b>	- run command on all files of a folder		147
<b>sfk runloop</b>	- run a command n times in a loop	151	
<b>sfk printloop</b>	- print some text many times		151
<b>sfk load</b>	- load file content for further use	232	
<b>sfk perline</b>	- run sfk command(s) per input text line		152
<b>sfk strings</b>	- extract strings from a binary file	154	
<b>sfk sort</b>	- sort text lines produced by another command		263
<b>sfk count</b>	- count text lines, filter identical lines	274	
<b>sfk head</b>	- print first lines of a file	239	
<b>sfk tail</b>	- print last lines of a file		239
<b>sfk linelen</b>	- tell length of string(s)		314

## search and compare

<code>sfk xfind</code>	- search in text and binary files using wildcards and simple expressions	114
<code>sfk xtext</code>	- search in text files only	124
<code>sfk xhexfind</code>	- search with hexdump output	124
<code>sfk extract</code>	- extract data from text and binary	125
<code>sfk find</code>	- search static text, without wildcards	109
<code>sfk ftext</code>	- search static text in text files only	109
<code>sfk hexfind</code>	- search static binary data	111
<code>sfk md5gento</code>	- create list of md5 checksums over files	95
<code>sfk md5check</code>	- verify list of md5 checksums over files	96
<code>sfk md5</code>	- calc md5 over a file, compare two files	97
<code>sfk pathfind</code>	- search PATH for location of a command	154
<code>sfk reflist</code>	- list fuzzy references between files	155
<code>sfk deplist</code>	- list fuzzy dependencies between files	157
<code>sfk dupfind</code>	- find duplicate files by content	248

## networking

<code>sfk httpserv</code>	- run an instant HTTP server. type "sfk httpserv -help" for help.	167
<code>sfk ftpserv</code>	- run an instant FTP server type "sfk ftpserv -help" for help.	168
<code>sfk ftp</code>	- instant anonymous FTP client	172
<code>sfk wget</code>	- download HTTP file from the web	199
<code>sfk web</code>	- send HTTP request to a server	197
<code>sfk tcpdump</code>	- print TCP conversation between programs	179
<code>sfk udpdump</code>	- print incoming UDP requests	180
<code>sfk udpsend</code>	- send UDP requests	183
<code>sfk ip</code>	- tell own machine's IP address(es). type "sfk ip -help" for help.	189
<code>sfk netlog</code>	- send text outputs to network, and/or file, and/or terminal	185
<code>sfk fromnet -h</code>	- receive and print network text	181
<code>sfk ping</code>	- ping multiple machines in one go	265

## scripting

<code>sfk script</code>	- run many sfk commands in a script file	257
<code>sfk label</code>	- define starting points within a script	261
<code>sfk call</code>	- call a sub function at a label	271
<code>sfk echo</code>	- print (coloured) text to terminal	159
<code>sfk color</code>	- change text color of terminal	163
<code>sfk setvar</code>	- put text into an sfk variable	294

<b>sfk storetext</b>	- store text in memory for later use	296
<b>sfk alias</b>	- create command from other commands	215
<b>sfk mkcd</b>	- create command to reenter directory	215
<b>sfk sleep</b>	- delay execution for milliseconds	238
<b>sfk pause</b>	- wait for user input	238
<b>sfk stop</b>	- stop sfk script execution	273
<b>sfk tee</b>	- split command output in two streams	262
<b>sfk tofile</b>	- save command output to a file	262
<b>sfk toterm</b>	- flush command output to terminal	262
<b>sfk for</b>	- repeat commands many times	322
<b>sfk loop</b>	- repeat execution of a command chain	323
<b>sfk cd</b>	- change directory within a script	214
<b>sfk getcwd</b>	- print the current working directory	214
<b>sfk require</b>	- compare version text	166
<b>sfk time -h</b>	- print current date and time	161
<b>development</b>		
<b>sfk bin-to-src</b>	- convert binary data to source code	134
<b>sfk make-random-file</b>	- create file with random data	162
<b>sfk fuzz</b>	- change file at random, for testing	196
<b>sfk sample</b>	- print example code for programming	250
<b>sfk inst</b>	- instrument c++ with tracing calls	153
<b>diverse</b>		
<b>sfk status</b>	- send colored status to the SFKTray Windows GUI utility for display	269
<b>sfk calc</b>	- do a simple instant calculation	271
<b>sfk random</b>	- create a random number	324
<b>sfk prompt</b>	- ask for user input	324
<b>sfk number</b>	- print number in diverse formats	267
<b>sfk xmlform</b>	- reformat xml for easy viewing	264
<b>sfk media</b>	- cut video and binary files	192
<b>sfk view</b>	- show results in a GUI tool	240
<b>sfk toclip</b>	- copy command output to clipboard	236
<b>sfk fromclip</b>	- read text from clipboard	237
<b>sfk env</b>	- search environment variables	260
<b>sfk version</b>	- show version of a binary file	164
<b>sfk ascii</b>	- list ISO 8859-1 ASCII characters	277
<b>sfk ascii -dos</b>	- list OEM codepage 850 characters	277
<b>sfk spell [-h]</b>	- phonetic spelling for telephone	265
<b>sfk cmd</b>	- print an example command	323
<b>sfk data</b>	- create random test data	244

<code>sfk ruler</code>	- measure console text width	246
<code>sfk license</code>	- print the SFK license text	
<code>sfk update</code>	- check for SFK updates	274
<b>help by subject</b>		
<code>sfk help select</code>	- how dirs and files are selected in sfk	297
<code>sfk help options</code>	- general options reference	278
<code>sfk help patterns</code>	- wildcards and text patterns within sfk	301
<code>sfk help chain</code>	- how to combine multiple commands	285
<code>sfk help var</code>	- how to use sfk variables and params	291
<code>sfk help shell</code>	- optimize the windows command prompt	283
<code>sfk help chars</code>	- about locale specific characters	304
<code>sfk help nocase</code>	- about case insensitive search	306
<code>sfk help unicode</code>	- about unicode file reading support	303
<code>sfk help colors</code>	- how to change result colors	275
<code>sfk help compile</code>	- how to compile sfk on any linux	312
<code>sfk help xe</code>	- for infos on xe specific features	334
<b>Alphabetical Index</b>		336

## 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.**

**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**

**The Apple Mac OS/X binaries are available by:**

`wget http://stahlworks.com/sfkmac`      **for current Intel based Macs**

`wget http://stahlworks.com/sfkmacold`      **for PowerPC based Macs**

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

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

**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
```

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

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

or on a Linux/Mac console, type:

```
wget http://othermachine:9090/sfkux
```

*further reading:*

httpserv tutorial on page 41,  
reference on page 167.

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 41,  
or the reference on page 168.

## 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 283.*

If you create a collection of batch files (e.g. through the "sfk alias" command on page 215) 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 275.

## 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 cannot read .zip, .tar.gz or .tar.bz2 contents and does not contain the high performance replace and xreplace commands.
- **SFK Base+XD**: these are the binaries you can download from stahlworks.com and SourceForge. They also contain a .zip file reading demo that reads the first 1000 bytes of every .zip file entry, and an xreplace demo that cannot write files.
- **SFK XE or Extended Edition**: this is the commercial edition of SFK available from stahlworks.com. It contains a high performance implementation of sfk replace and the sfk xreplace command. Furthermore it can search .zip, .tar.gz and .tar.bz2 file contents directly with several commands.

Some Linux distributions allow installation of SFK via their package managers, which 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 1.8.2
```

## 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, 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 the words "new" and "scm":

```
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 85.*

## 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 86.*

## Find a filename quickly in the current directory tree

You are standing within a folder and know that a file having foo somewhere in it's 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 it's **path** and "sub2" in it's **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 243.*



## 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 86.*

## 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 147.*

## 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 -sincdiff step1 step2 +run "winmerge $qsince $qfile"
```

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

remarks:

- now we don't use -sincdir but -sincdiff 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 147.*

## Rename files quickly using patterns

I have a folder with tv recordings like:

```
Cable2Newdyne-07272011-1337-World News.mts
Cable9Kenton-08022013-1025-Market News.mts
Cable_Gennex-07232013-1025-Market News.mts
Cable_Weldyne-05162014-1025-Market News.mts
Channel5Framdyne-06162013-0653-The Larry Scott News.mts
Channel_Eltree-09232014-1230-World News.mts
News9Dancar-04082012-0117-Technology News.mts
News_Cancar-04122011-0337-The Alexander Johnson Talk.mts
News_Danmond-04192012-1906-Donald Johnson News.mts
...
```

and need to change many of those names quickly.

## Change all names having a pattern

Everything with a time "1025" is not really "Market News" but "The Foo Show". Just because I programmed recording a few minutes before the Foo Show starts it got the wrong name. To fix this I go into folder "recordings" and then type:

```
sfk rename . "/-1025-*.mts/[part1]The Foo Show[part3]/"
```

remarks:

- first parameter "." means "do this on all files in current folder". With `sfk rename` files in sub folders are *not* included by default.
- second parameter is a search and replace pattern in the form `/fromtext/totext/` surrounded by separator slashes. The fromtext is searched and then replaced by the totext.
- the fromtext contains three parts: a literal "-1025-", a wildcard "\*" and another literal ".mts". this matches any filename having "-1025-" and ".mts" somewhere.
- the totext can use these parts. `[part1]` simply means to fill in the "-1025-" so we don't have to type it again, same for `[part3]`.