The Swiss File Knife Book

Version 1.9.9

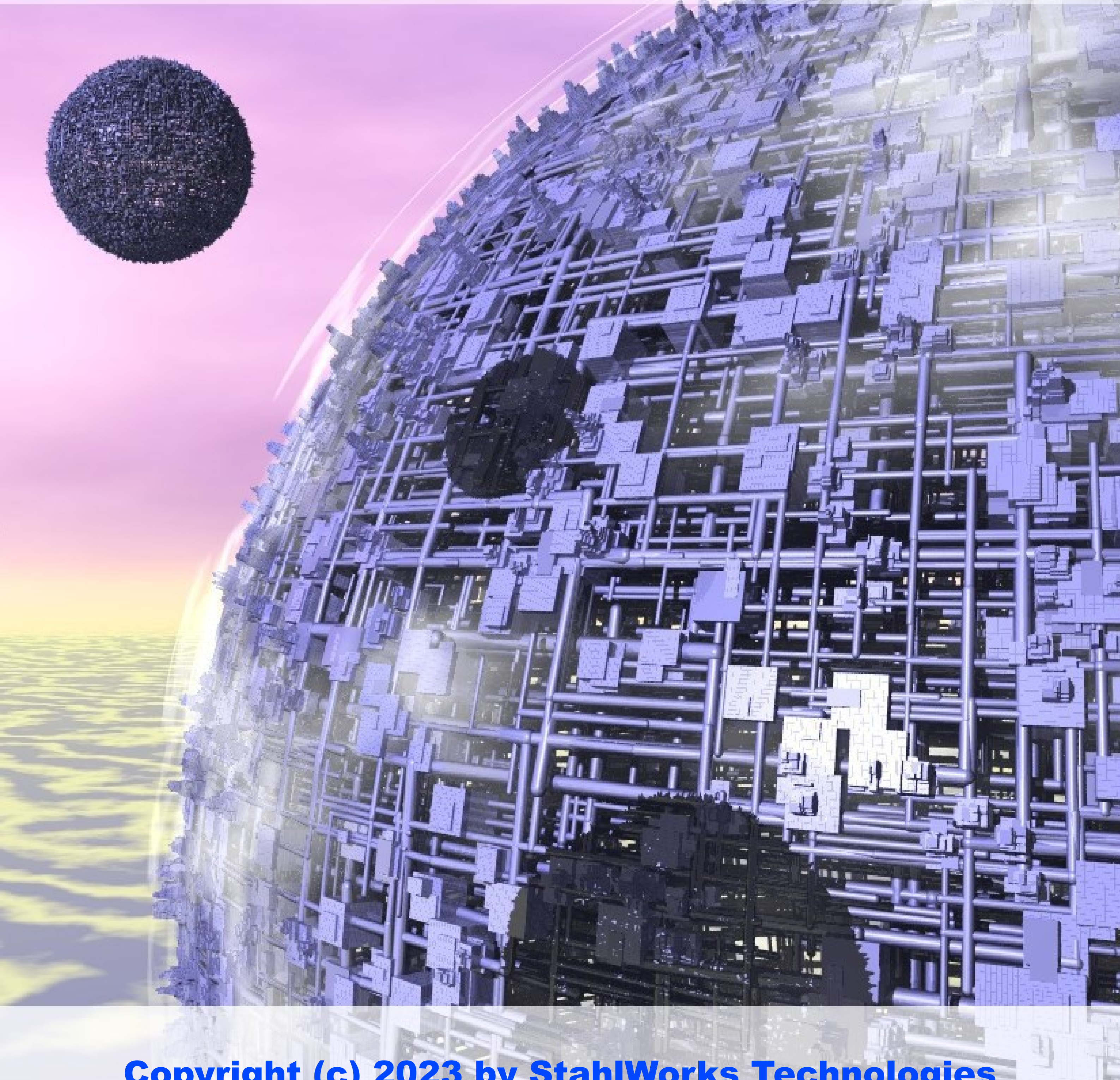

Table of contents

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 9
How to get the SFK tool running instantly 10
The different editions of SFK: OSE, Base, XE 13

Tutorial 14

File handling

List all files of a folder, and all sub folders 14
List only selected files in selected sub folders 15
List files using wildcards 16
List the latest or biggest files 16
Find a file quickly in the current directory tree 17
List different files between two folders 18
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 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 51

Backup and transfer of folders
A script to backup files to an USB drive 52
Sync holiday snapshot name changes to USB drive 59
Copy a project folder to a new version 60
Upload project files to a build server 61
Optimizing file uploads by modification date 63

Xed detailed examples 64
reformat comma separated data 65
convert fixed column data to CSV 66
convert CSV data to XML data 67
convert XML data to CSV data 69
cleaning up a translation file 71
extract two letter words from text 72

Large script examples
Wiki markup text to HTML conversion
using sfk script and xed 73

HTTP Scripting and Test Automation
using sfk script, call, web, perline, xex 78

Filling an XML file with program meta data
using sfk script, xex, version, list, calc,
filter, setvar, getvar, xed 83

A detailed perline example 90

Script creation and debugging tips 93

Command Reference

General infos 95

Windows/Linux/Mac syntax differences 97

file system

sfk list	- list directory tree contents. list latest, oldest or biggest files. list directory differences. list zip jar tar gz bz2 contents.	98
sfk filefind	- find files by filename	263
sfk treesize	- show directory size statistics	120
sfk copy	- copy directory trees additively	222
sfk sync	- mirror tree content with deletion	222
sfk rename	- flexible multi file rename	226
sfk partcopy	- copy part from a file into another one	207
sfk mkdir	- create directory tree	232
sfk delete	- delete files and folders	117
sfk deltree	- delete whole directory tree	117
sfk deblank	- remove blanks in filenames	170
sfk space [-h]	- tell total and free size of volume	218
sfk filetime	- tell times of a file	220
sfk touch	- change times of a file	255
sfk index	- create index file(s) for fast lookup	104
sfk name	- lookup file names using index files	106
sfk fixfile	- change bad filenames and file times	292
sfk setbytes	- set bytes at offset within a file	350

compression

sfk zip	- create zip file from folder	354
sfk zipto	- zip selected file list	358
sfk unzip	- list or extract zip file	359
sfk checkzip	- verify zip file content	362

conversion

sfk oload	- load office file content as text	252
sfk lf-to-crlf	- convert from LF to CRLF line endings	118
sfk crlf-to-lf	- convert from CRLF to LF line endings	120
sfk detab	- convert TAB characters to spaces	115
sfk entab	- convert groups of spaces to TAB chars	116

sfk scantab - list files containing TAB characters 116
sfk split - split large files into smaller ones 205
sfk join - join small files into a large one 206
sfk csvtotab - convert .csv data to tab separated 253
sfk tabtocsv - convert tab separated to .csv format 254
sfk encode - convert data to base64 or hex format 344
sfk decode - decode base64, hex or url format 344
sfk wtoa - Windows: convert wide chars to Ansi 345
sfk wtou - convert wide chars to UTF-8 347
sfk utoa - convert UTF-8 text to Ansi 348
sfk hexdump - create hexdump from a binary file 193
sfk hextobin - convert hex data to binary 267
sfk hex - convert decimal number(s) to hex 281
sfk dec - convert hex number(s) to decimal 281
sfk chars - print chars for a list of codes 266
sfk bin-to-src - convert binary to source code 148
sfk uuencode - encode binary files as plain text 363
sfk uudecode - decode binary files from plain text 364

text processing
sfk filter - search, filter and replace text data 149
sfk ofilter - filter text from an office file 159
sfk xed - edit stream text using expressions 243
sfk xex - extract from stream text using expressions 247
sfk addhead - insert string at start of text lines 160
sfk addtail - append string at end of text lines 160
sfk patch - change text files through a script 161
sfk snapto - join many text files into one file 113
sfk joinlines - join text lines split by email client 115
sfk inst - instrument c++ sourcecode with tracing calls 168
sfk replace - replace words in binary and text files 236
sfk xreplace - replace in files using expressions 141
sfk hexfind - find words in binary files, showing hexdump 124
sfk run - run command on all files of a folder 162
sfk runloop - run a command n times in a loop 166
sfk printloop - print some text many times 166
sfk load - load file content for further use 251
sfk perline - run sfk command(s) per input text line 167
sfk strings - extract strings from a binary file 169
sfk sort - sort text lines produced by another command 285
sfk count - count text lines, filter identical lines 300
sfk head - print first lines of a file 259

sfk tail - print last lines of a file 259
sfk linelen - tell length of string(s) 343

search and compare

sfk xfind - search in text files using wildcards and simple expressions 127
sfk ofind - search in office files .docx .xlsx .ods 137
sfk xhexfind - search with hexdump output 138
sfk extract - extract data from text and binary 139
sfk find - search static text, without wildcards 122
sfk hexfind - search static binary data 124
sfk md5gento - create list of md5 checksums over files 108
sfk md5check - verify list of md5 checksums over files 109
sfk md5 - calc md5 over a file, compare two files 110
sfk pathfind - search PATH for location of a command 170
sfk reflist - list fuzzy references between files 170
sfk deplist - list fuzzy dependencies between files 172
sfk dupfind - find duplicate files by content 268

networking

sfk httpserv - run an instant HTTP server. 182
type "sfk httpserv -help" for help.
sfk ftpserv - run an instant FTP server 184
type "sfk ftpserv -help" for help.
sfk ftp - instant FTP client 188
sfk wget - download HTTP file from the web 216
sfk web - send HTTP request to a server 213
sfk tcpdump - print TCP conversation between programs 195
sfk udpdump - print incoming UDP requests 196
sfk udpsend - send UDP requests 199
sfk webproxy - http proxy for traffic analysis 366
sfk help tcp - tcp toolkit to write test scripts 366
sfk ip - tell own machine's IP address(es). 205
type "sfk ip -help" for help.
sfk netlog - send text outputs to network, and/or file, and/or terminal 201
sfk fromnet -h - receive and print network text 198
sfk ping - ping multiple machines in one go 288
sfk pingdiff - find ip of new devices 289

scripting

sfk batch - run many sfk commands in a script file 280

sfk script - run a script file directly 277
sfk label - define starting points within a script 282
sfk call - call a sub function at a label 296
sfk echo - print (coloured) text to terminal 174
sfk color - change text color of terminal 178
sfk setvar - put text into an sfk variable 322
sfk storetext - store text in memory for later use 324
sfk alias - create command from other commands 234
sfk mkcd - create command to reenter directory 233
sfk sleep - delay execution for milliseconds 258
sfk pause - wait for user input 258
sfk stop - stop sfk script execution 297
sfk tee - split command output in two streams 284
sfk tofile - save command output to a file 284
sfk toterm - flush command output to terminal 284
sfk for - repeat commands many times 352
sfk loop - repeat execution of a command chain 352
sfk cd - change directory within a script 232
sfk getcwd - print the current working directory 232
sfk require - compare version text 181
sfk time -h - print current date and time 176

development
sfk bin-to-src - convert binary data to source code 148
sfk make-random-file - create file with random data 178
sfk fuzz - change file at random, for testing 212
sfk sample - print example code for programming 270
sfk inst - instrument c++ with tracing calls 168

diverse
sfk status - send colored status to the SFKTray Windows GUI utility for display 293
sfk calc - do a simple instant calculation 295
sfk random - create a random number 353
sfk prompt - ask for user input 353
sfk number - print number in diverse formats 291
sfk xmlform - reformat xml for easy viewing 285
sfk jsonform - reformat json for easy viewing 286
sfk video - how to edit video files 208
sfk view - show results in a GUI tool 260
sfk toclip - copy command output to clipboard 256
sfk fromclip - read text from clipboard 257

sfk env - search environment variables 281
sfk version - show version of a binary file 179
sfk ascii - list ISO 8859-1 ASCII characters 302
sfk ascii -dos - list OEM codepage 850 characters 302
sfk spell [-h] - phonetic spelling for telephone 287
sfk cmd - print an example command 353
sfk ruler - measure console text width 266
sfk license - print the SFK license text
sfk update - check for SFK updates 299

help by subject
sfk help office - how to search in office files 330
sfk help select - how dirs and files are selected in sfk 325
sfk help options - general options reference 303
sfk help patterns - wildcards and text patterns within sfk 329
sfk help chain - how to combine multiple commands 311
sfk help var - how to use sfk variables and params 318
sfk help shell - optimize the windows command prompt 309
sfk help chars - about locale specific characters 333
sfk help nocase - about case insensitive search 335
sfk help unicode - about unicode file reading support 332
sfk help colors - how to change result colors 300
sfk help compile - how to compile sfk on any linux 341
sfk help xe - for infos on xe specific features 368

Alphabetical Index 370

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 64-bit i686 systems

   wget http://stahlworks.com/sfkux32    for 32-bit i686 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 Intel 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 42,
reference on page 182.

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 184.

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 309.

If you create a collection of batch files (e.g. through the "sfk alias" command on page 234) 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 300.

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     1.9.8
```

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 97.

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* 98.**

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 263.

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* 98.**

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 162.

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 -sincediff step1 step2 +run "winmerge $qsince $qfile"
```

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

- **now we don't use** -sincedir **but** -sincediff **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 162.