RFTools Manual 1.0.0

RFtools is a collection of utilities for handling filesequences in the unix commandline.

Filesequences are list of files separated by changing filenumber commonly used in film and visual effects work.

– example.0000.dpx
– example.0001.dpx
– example.0002.dpx

Instead of listing all individual files filesequence tools allow to represent filelists as single line entries.
For example when used together with 'ls' command.

> ls
example.000001.dpx
example.000002.dpx
example.000003.dpx
...
example.100000.dpx

With rfpack

> ls | rfpack
example.[0-100000@@@@@@].dpx

From a list of individual files rfpack creates compressed oneliner in ***'rf filesequence format'.*** All the filesequence tools except rfbuild work strictly manipulating lists of text and do not care about the actual files or their contents.

rfunpack is the opposite of rfpack, it expands filelists from ***'rf filesequence format'***

Here are some examples which show as well the basic properties of ***'rf filesequence format'***

> rfunpack filename.[0-2@].extension
filename.0.extension
filename.1.extension
filename.2.extension

> rfunpack sequence_in_two_parts.[0-1@,8-9@].extension
sequence_in_two_parts.0.extension
sequence_in_two_parts.1.extension
sequence_in_two_parts.8.extension
sequence_in_two_parts.9.extension

> rfunpack four_digit_padding.[20-22@@@@].dpx
four_digit_padding.0020.dpx
four_digit_padding.0021.dpx
four_digit_padding.0022.dpx

> rfunpack every_second_frame.[0-6x2@@@].exr
every_second_frame.000.exr
every_second_frame.002.exr
every_second_frame.004.exr
every_second_frame.006.exr

> rfunpack descending_and_negative.[2--4x-1@@@@].png
descending_and_negative.0001.png
descending_and_negative.0000.png
descending_and_negative.-001.png
descending_and_negative.-002.png

Combining tools

The preferred way to combine tools is to pipe them together like in the previous example

> ls | rfpack

Filesequence tools can also take in filenames or sequence descriptions as arguments.

> rfpack test.0.dpx test.1.dpx test.2.dpx
test.[0-2@].dpx

> rfunpack test.[0-2@].dpx
test.0.dpx
test.1.dpx
test.2.dpx

> rfpack *
(returns all the sequences in directory via shell filename expansion.
Similar to 'ls | rfpack', but only the latter will work with directories with large amount of files)

Also mixed input is possible, piped in text is processed before the arguments

> echo "test.[0-2@@].dpx" | rfunpack.py test.[3-4@@].dpx
test.00.dpx
test.01.dpx
test.02.dpx
test.03.dpx
test.04.dpx

Mix and match with the standard unix tools.
For finding filesequences with the word 'foo' in their name for subdirectory bar.

> find ./bar -type f | rfpack | grep ”foo”

Combatibility with other software

Many programs have varying ways of descriping filesequences similar to filesequence tools. 'rformat' provides combatibility with some of these as well as provides different ways listing out filesequences.

To output in 'printf' format

> ***> rfformat example.[0-3@@@@].exr -o filmlight***
> ***example.%.4F.exr:0-3***
>
> or
>
> ***> ls | rfpack | rfformat -o rv***
> ***(lists sequences in dir in 'rv' format)***

-o or ***--output*** supports also custom user defined formats. Any string that is not known preset output format is taken as prototype for custom format. Special wildcards are substituted by features of the sequence at hand.

To print simpler prototype where number is substituted with ***'#'***

> ***> rfformat padding_four.[0-2@@@@].exr padding_two.[3-5@@].dpx -o "%h%#%t"***
> ***padding_four.####.exr***
> ***padding_two.##.dpx***

Print more detailed information of all sequences in subdirs

> ***> find . | rfpack | rfformat --fps 25 -o "%f frames: %s-%e start tc: %<"***
> ***./shot01/shot01.[90000-90013@@@@@].png frames: 90000-90013 start tc: 01:00:00:00***
> ***./shot02/shot02.[90014-90043@@@@@].png frames: 90014-90043 start tc: 01:00:00:14***
> ***./shot03/shot03.[90044-90050@@@@@].png frames: 90044-90050 start tc: 01:00:01:19***

Timecodes are always translated from framenumbers.
Note when using timecode with any filesequence tool you need to always explicitly set ***--fps.***

Note that possible typos cause unexpected output

> ***> find . | rfpack | rfformat -o prinft***
> ***prinft***
> ***prinft***
> ***...***

For full list of custom output format wildcards and preset output formats see reference documentation.

-i or ***--input*** can be used for reading filesequence descriptions from other programs

> ***> cat baselight_sequences.txt | rfformat --input filmlight --output rv > rv_sequences.txt***
> (creates file called rv_sequences.txt, which contains translated sequence names from a list generated by filmlight tools)

Editing sequences

rfedit provides tools for editing filesequence descriptions.
It does not touch the filesequences themselves, but only filters and modifies the descriptions.

Let's assume we have six sequences in six directories

```
> find . | rfpack | rfformat -o "%f has %l file(s)"
./five_files/five_files.[1-5@@@@].dpx has 5 file(s)
./four_files/four_files.[1-4@@@@].dpx has 4 file(s)
./one_file/one_file.[1-1@@@@].dpx has 1 file(s)
./six_files/six_files.[1-6@@@@].dpx has 6 file(s)
./three_files/three_files.[1-3@@@@].dpx has 3 file(s)
./two_files/two_files.[1-2@@@@].dpx has 2 file(s)
```

To filter out sequences that contain less than 5 frames

```
> find . | rfpack | rfedit --min 5
./five_files/five_files.[1-5@@@@].dpx
./six_files/six_files.[1-6@@@@].dpx
```

Let's move the description of two longest sequences to start at frame 0

```
> find . | rfpack | rfedit --min 5 --start 0
./five_files/five_files.[1-5@@@@].dpx -> ./five_files/five_files.[0-4@@@@].dpx
./six_files/six_files.[1-6@@@@].dpx -> ./six_files/six_files.[0-5@@@@].dpx
```

Now rfedit shows both the original filename description as well as the new filename separated by **' -> '.**
To actually perform the renaming of files this output needs to be fed to rfbuild as seen in the chapter concerning rfbuild

Here are few other common uses of rfedit

reconstruct input sequences into one combined sequence

```
> find . | rfpack | rfedit --reconstruct "combined_sequence.[0-1000@@@@].dpx"
./five_files/five_files.[1-5@@@@].dpx -> combined_sequence.[0-4@@@@].dpx
./four_files/four_files.[1-4@@@@].dpx -> combined_sequence.[5-8@@@@].dpx
./one_file/one_file.[1-1@@@@].dpx -> combined_sequence.[9-9@@@@].dpx
./six_files/six_files.[1-6@@@@].dpx -> combined_sequence.[10-15@@@@].dpx
./three_files/three_files.[1-3@@@@].dpx -> combined_sequence.[16-18@@@@].dpx
./two_files/two_files.[1-2@@@@].dpx -> combined_sequence.[19-20@@@@].dpx
```

Note that the target sequence is defined to be length of 1001 frames, but it is truncated to the combined length of source files (0-20). If target sequence of ***--reconstruct*** is shorter than source lengths then that will be the limiting factor.

The sequences are in alphabetical order. To reorder the clips from shortest to longest before reconstructing the new sequence we can use --reorder. Reorder reads input sequences in the order of quoted list.

```
> find . | rfpack | rfedit --reorder "3,6,5,2,1,4" --reconstruct "combined.[0-1000@@@@].dpx"
./one_file/one_file.[1-1@@@@].dpx -> combined.[0-0@@@@].dpx
./two_files/two_files.[1-2@@@@].dpx -> combined.[1-2@@@@].dpx
./three_files/three_files.[1-3@@@@].dpx -> combined.[3-5@@@@].dpx
./four_files/four_files.[1-4@@@@].dpx -> combined.[6-9@@@@].dpx
./five_files/five_files.[1-5@@@@].dpx -> combined.[10-14@@@@].dpx
./six_files/six_files.[1-6@@@@].dpx -> combined.[15-20@@@@].dpx
```

Take one frame sample from start of each sequence.

```
> find . | rfpack | rfedit --head 1 | rfunpack
./five_files/five_files.0001.dpx
./four_files/four_files.0001.dpx
./one_file/one_file.0001.dpx
./six_files/six_files.0001.dpx
./three_files/three_files.0001.dpx
./two_files/two_files.0001.dpx
```

Move sequences to start at frame representing 10:00:00:00 (--fps needs to be set)

```
> find . | rfpack | rfedit --start 10:00:00:00 --fps 25
./five_files/five_files.[1-5@@@@].dpx -> ./five_files/five_files.[900000-900004@@@@].dpx
./four_files/four_files.[1-4@@@@].dpx -> ./four_files/four_files.[900000-900003@@@@].dpx
./one_file/one_file.[1-1@@@@].dpx -> ./one_file/one_file.[900000-900000@@@@].dpx
./six_files/six_files.[1-6@@@@].dpx -> ./six_files/six_files.[900000-900005@@@@].dpx
./three_files/three_files.[1-3@@@@].dpx -> ./three_files/three_files.[900000-900002@@@@].dpx
./two_files/two_files.[1-2@@@@].dpx -> ./two_files/two_files.[900000-900001@@@@].dpx
```

Filter 1 second handles from start and end of 24 fps sequence

```
> rfedit seq_with_handles.[348934-349023@@@@@@@].ari --trim "01:00" "01:00" --fps 24
seq_with_handles.[348958-348999@@@@@@@].ari -> seq_with_handles.
[348958-348999@@@@@@@].ari
```

Applying changes to files

'rfbuild' is tool for moving, copying, linking the actual files based on the output of other filesequence tools.

Creating combined sequence linking to all individual sequences in subdirectories

> ***> find . \| rfpack \| rfedit --reconstruct "combined.[0-1000@@@@].dpx" \| rfbuild --ln ./new_seq -x***

This will try to build combined sequence into subdirectory ***'new_seq'.***
If target directory does not exist -p flag can be added so that missing directories are automatically created.

If ran without ***-x (--execute)*** rfbuild will show ***'dry-run'*** preview of actions it's about to do.
It is highly recommended to run it without ***-x*** always before actually comitting actions.

MV:

Move (rename) 0 based sequence to 1 based in the same directory.

> ***> ls \| rfpack \| rfedit --start 1 \| rfbuild --mv . -x***

--mv is built so that it reorganizes move operations to avoid overwriting files. This build will usually happen backwards so that moving files won't overwrite original files. If we try to run the same build with copy ***(--cp)*** it will fail because of existing files. ***--force flag*** can be used to force build to happen, but it will corrupt the sequence. It is usually better idea to create new sequences to empty directories when possible to avoid conflicts of filenames.

RV:

rfbuild also supports launching selected sequences in Tweaksoftware RV. If you have rv installed so that you can launch it from shell with the command ***'rv'*** you should be able to use this feature as well. Alternatively you can set RV_EXECUTABLE_PATH environment variable to point to rv executable.

To find all dpx sequences in all subdirectories of current directory and load them into rv.

> ***> find . -type f -name "*.dpx" \| rfpack \| rfbuild --rv "-fullscreen"***

Flags following ***--rv*** will be passed down to rv executable ***(here -fullscreen).*** Use **--** to skip passing any flags.

--rv reads always the original filenames, so if you have ***'rfedit'*** in front of ***'rfbuild --rv'*** only the filtering options of edit have effect, but renaming operations will have no effect.

CMD:

Custom commands can be launched for each file in filesequence with ***--cmd.***
--cmd takes two arguments, first one is path to the executable and second are the arguments passed to this command. Two special wildcards ***%modified*** and ***%original*** can be used to refer to the filenames in the filesequence.

To generate 25 empty files with unix command ***'touch'***

> ***> rfbuild empty.[1-25@@@@].txt --cmd touch "%original"***

This monster will create thumbnail of first frame of each dpx sequence.
It uses imagemagick ***'convert'*** for rendering.

Arguments passed down to ***--cmd*** executable must be quoted.

> ***> find . -type f -name "*.dpx" \| rfpack \| rfedit --head 1 --reconstruct "thumb.[0-1000@@@@].dpx" \| rfbuild --cmd convert "%original -resize 120x120 %modified***"

About timecodes

Most filesequence tools support using timecodes instead of framenumbers.
Tools make no assumptions about the framerate so it needs to be set via ***--fps*** flag or setting global environment variable ***RF_FPS***. At the moment there is only support for non-dropframe fullframe timecodes ***(no 29.97 etc.).***

Timecodes don't have to be fully defined, shorter timecodes are assumed to represent last digits of full timecode. Following formats are allowed.

> ***10:00:00:00 = 10 hours, 0 minutes, 0 secs, 0 frames = frame 900000 (@ 25fps)***
> ***59:58:00 = 59 minutes, 58 seconds, 0 frames = frame 86352 (@ 24fps)***
> ***01:00 = 1 sec, 0 frames = frame 25 (@ 25fps)***

Usually timecodes are interpreted to be based on frame 0, but it is also possible to offset timecode zero point with ***--tcstart*** flag. Some examples

Straight mapping

> ***> rfformat test.[900000-900001@@@@@@@@].dpx --fps 25 -o "range %<-%>"***
> ***range 10:00:00:00-10:00:00:02***

Offset timecode start point by one minute (1500 frames)

> ***> rfformat test.[900000-900001@@@@@@@@].dpx --fps 25 --tcstart 1500 -o "range %<-%>"***
> ***range 09:59:00:00-09:59:00:02***

Offset timecode start point by 10 hours mapping 10:00:00:00 -> frame 0

> ***> rfformat test.[900000-900001@@@@@@@@].dpx --fps 25 --tcstart 10:00:00:00 -o "range %<-%>"***
> ***range 00:00:00:00-00:00:00:02***

Note that there is difference with timecode range and frame range.
Timecode range runs over by one frame compared to frame range so that consecutive clips have the same end and start timecode.

> ***> rfformat first_clip.[0000-0024@@@@].dpx second_clip.[0025-50@@@@].dpx***
> ***--fps 25 -o "%h framerange %s-%e, tc range %<-%>"***
> ***first_clip. framerange 0-24, tc range 00:00:00:00-00:00:01:00***
> ***second_clip. framerange 25-50, tc range 00:00:01:00-00:00:02:01***

RFPACK

NAME

rfpack – Finds sequences in a list of filenames.

SYNOPSIS

rfpack [-h] [-p] [-s] [-1] [-n] [-i] [string ...]

DESCRIPTION

rfpack reads in list of strings (usually filenames) both as arguments and from stdin.
It detects sequences and returns single entry for each filesequence.

OPTIONS

-h, --help	Display simple help.
-p, --passthrough	Pass through lines that are not part of any sequence unchanged.
-s, --sort	Sort files based on their name and framenumbering before detecting filesequences
-1, --single	Try to force single files containing number to be treated as 1 frame sequences
-n, --negative	Allow negative frame numbers. ***'test-20.dpx'*** can be frame -20 of sequence ***'test###.dpx'*** or frame 20 of sequence ***'test-##.dpx***'
-i, --invert	List missing files in a sequence rather than the files. Sequences without missing files will be omitted.

EXAMPLES

Following creates filesequence of three filenames

rfpack file.0001.dpx file.0002.dpx file.0003.dpx
out: file.[1-3@@@@].dpx

Detect filesequence with stepping frame numbers and two number padding

rfpack file.00.dpx file.04.dpx file.08.dpx
out: file.[0-8x4@@].dpx

Descending filesequence with negative framenumbers

rfpack -n file5.dpx file0.dpx file-5.dpx
out: file[5--5x-5@].dpx

Multipart filesequence with missing frames

rfpack file.0001.dpx file.0002.dpx file.0003.dpx file.0005.dpx file.0007.dpx file.0009.dpx
out: file.[1-3@@@@,5-9x2@@@@].dpx

Inverse of multipart filesequence with missing frames

rfpack -i file.0001.dpx file.0002.dpx file.0003.dpx file.0005.dpx file.0007.dpx file.0009.dpx
out: file.[4-4@@@@,6-6@@@@,8-8@@@@].dpx

Use in combination with standard unix tools

ls | sort | rfpack
find . -name "*.dpx" | rfpack

RFUNPACK

NAME

rfunpack – Expand sequence description back into list of files.

SYNOPSIS

rfunpack [-h] [filesequence_string ...]

DESCRIPTION

rfunpack reads filesequence descriptions such as ***example.[0-10@@].dpx*** and prints out
a list of individual filenames in that sequence. If the input is in the ***'original -> modified***' filesequence format
created by ***rfedit(1)*** both the original filename and the modified filenames are expanded.

OPTIONS

-h, --help Display simple help.

EXAMPLES

Unpack the individual files in sequence:

```
> rfunpack file.[1-3@@@@].dpx
file.0001.dpx
file.0002.dpx
file.0003.dpx
```

Print edited output:

```
> rfpack file.001.dpx file.002.dpx file.003.dpx | rfedit --start 0 | rfunpack
file.0001.dpx -> file.0000.dpx
file.0002.dpx -> file.0001.dpx
file.0003.dpx -> file.0002.dpx
```

RFEDIT

NAME

rfedit – Edit or filter filesequences.

SYNOPSIS

rfedit [-h] [-1] [--fps framerate] [--tcstart frame] [-TtHLMmposevrxR]
[filesequence _ string ...]

DESCRIPTION

rfedit manipulates and filters filesequence strings. If editing causes filenames to change rfedit can output pair of filesequence strings in the format ***'original -> modified'.*** The output of rfedit can be fed to rfbuild which will modify the actual files. Edit flags will be processed in the order they are provided.

OPTIONS (general)

-h, --help display simple help

-1, --single Always return the single-column output of modified filenames, omitting possible original filenames.

--fps set global framerate (frames per second). Filesequence tools make no assumptions about framerate of filesequence, so it needs to be explicitly set each time timecodes are used. This is usually when frame numbers are displayed as timecode or length of sequence is displayed as timecode.

--tcstart set global start of timecode. Timecodes are usually mapped so that frame 0 matches timecode 00:00:00:00. This option can be used to remap timecode 00:00:00:00 to different frame.

OPTIONS (filtering)

Filtering options remove entries from input list when match is found.
This can mean removing frames from filesequence string or omitting whole sequences.

-T, --truncate start end Only frames residing inside provided range will be kept, all the other frames are filtered out.

-t, --trim head tail Trim each sequence by provided length from head and tail. Note that there is difference if arguments to ***--trim*** are provided as frames or as timecode. ***--trim 5 5*** will remove five first and last filenames of sequence, but ***--trim 01:00 01:00*** will remove one second worth of frames from start and end of sequence. This means that if sequence skips frames the amount of frames removed will be less than 25.

-H, --head frames Take N frames from the start of each sequence.

-L, --tail frames Take N last frames of each sequence.

-M, --max frames Only list sequences that are shorter than N frames.

-m, --min frames	Only list sequences that are longer than N frames.
-R, --reorder "3,2,1,4-"	Reorder input sequences based on a comma separated list. List can also contain ranges marked with ***'-'.*** To reverse 3 first sequences use ***"3,2,1,4-".*** To reverse 9 sequences use ***"9-1"*** or simply ***"9-"***

OPTIONS (modifying)

Modifying options change sequence name or numbering.

-p, --padding padding	Force custom padding to all clips.
-o, --offset frames	Offset clip numbering by N frames. Negative frame amount also accepted. ***--offset -1*** will move sequence from 1-10 to 0-9.
-s, --start frame	Offset sequence to start from provided frame.
-e, --end frame	Offset sequence to end at provided frame.
-S, --replace search replace	Search for string A and when found replace with string B. Leave filenumbering untouched.
-i, --ireplace search replace	Case insensitive search and replace.
-v, --reverse	Reverse the order and direction of clips in a sequence.
-r, --reconstruct new_seq	Map input sequences to a new sequence. Can be used to merge multiple input sequences to one long sequence.
-x, --removegaps	Remove gaps from a sequence by moving frames after the gap earlier.

EXAMPLES

Let's assume we have following sequences available

```
find * | rfpack
five_files/five_files.[1-5@@@@].dpx
four_files/four_files.[1-4@@@@].dpx
one_file/one_file.[1-1@@@@].dpx
six_files/six_files.[1-6@@@@].dpx
three_files/three_files.[1-3@@@@].dpx
two_files/two_files.[1-2@@@@].dpx
```

Filter out under 5 frame sequences

```
find * | rfpack | rfedit --min 5
five_files/five_files.[1-5@@@@].dpx
six_files/six_files.[1-6@@@@].dpx
```

Offset all sequences start at frame corresponding to timecode 10:00:00:00 at 25 fps

```
find * | rfpack | rfedit --fps 25 --start 10:00:00:00
five_files/five_files.[1-5@@@@].dpx -> five_files/five_files.[900000-900004@@@@].dpx
four_files/four_files.[1-4@@@@].dpx -> four_files/four_files.[900000-900003@@@@].dpx
one_file/one_file.[1-1@@@@].dpx -> one_file/one_file.[900000-900000@@@@].dpx
six_files/six_files.[1-6@@@@].dpx -> six_files/six_files.[900000-900005@@@@].dpx
three_files/three_files.[1-3@@@@].dpx -> three_files/three_files.[900000-900002@@@@].dpx
two_files/two_files.[1-2@@@@].dpx -> two_files/two_files.[900000-900001@@@@].dpx
```

Make sequences into one combined sequence

```
find * | rfpack | rfedit --reconstruct combined_sequence.[0-1000@@@@].dpx
five_files/five_files.[1-5@@@@].dpx -> combined_sequence.[0-4@@@@].dpx
four_files/four_files.[1-4@@@@].dpx -> combined_sequence.[5-8@@@@].dpx
one_file/one_file.[1-1@@@@].dpx -> combined_sequence.[9-9@@@@].dpx
 six_files/six_files.[1-6@@@@].dpx -> combined_sequence.[10-15@@@@].dpx
three_files/three_files.[1-3@@@@].dpx -> combined_sequence.[16-18@@@@].dpx
two_files/two_files.[1-2@@@@].dpx -> combined_sequence.[19-20@@@@].dpx
```

Reorder sequences manually from shortest to longest

```
find * | rfpack | rfedit --reorder "3,6,5,2,1,4"
one_file/one_file.[1-1@@@@].dpx
two_files/two_files.[1-2@@@@].dpx
three_files/three_files.[1-3@@@@].dpx
four_files/four_files.[1-4@@@@].dpx
five_files/five_files.[1-5@@@@].dpx
six_files/six_files.[1-6@@@@].dpx
```

Change extension to .jpg

```
find * | rfpack | rfedit --replace ".dpx" ".jpg"
one_file/one_file.[1-1@@@@].dpx -> one_file/one_file.[1-1@@@@].jpg
two_files/two_files.[1-2@@@@].dpx -> two_files/two_files.[1-2@@@@].jpg
three_files/three_files.[1-3@@@@].dpx -> three_files/three_files.[1-3@@@@].jpg
four_files/four_files.[1-4@@@@].dpx -> four_files/four_files.[1-4@@@@].jpg
five_files/five_files.[1-5@@@@].dpx -> five_files/five_files.[1-5@@@@].jpg
six_files/six_files.[1-6@@@@].dpx -> six_files/six_files.[1-6@@@@].jpg
```

RFFORMAT

NAME

rfformat - read in and print out filesequences in different styles

SYNOPSIS

rfformat [-h] [--fps framerate] [--tcstart frame] [-i format] [-o for-
mat] [filesequence_string ...]

DESCRIPTION

rfformat translates filesequence strings between different styles. Many programs have their own way of representing filesequences with strings. rfformat can convert many of these into native filesequence_tools_format and also export filesequences for other programs to read.

rfformat also supports free format output, which can be used for custom printouts of filesequences. See more in FORMATS.

OPTIONS

-h, --help display simple help

--fps set global framerate (frames per second). Filesequence tools make no assumptions about framerate of filesequence, so it needs to be explicitly set each time timecodes are used. This is usually when frame numbers are displayed as timecode or length of sequence is displayed as timecode.

--tcstart set global start of timecode. Timecodes are usually mapped so that frame 0 matches timecode 00:00:00:00. This option can be used to remap timecode 00:00:00:00 to different frame.

-i, --input set input format. If not set filesequence_tools_format is expected. See FORMATS for more information.

-o, --output set output format. If not set filesequence_tools_format is printed out. See FORMATS for more information.

FORMATS (in/out)

The following formats are supported both by --input and --output.

printf printf is subset of standard printf format string. It only supports ***%d*** wildcard. String is followed by space and numbers implying start and end frame of sequence. Start and end can be separated with space or any non-digit character. Examples:

'file.%04d.dpx 0 10' frames 0 to 10 with four digit padding
'file.%d.dpx 0-10' frames 0 to 10 with one digit padding

filmlight filesequence format used by filmlights software such as baselight. Similar to ***printf,*** but uses notation ***%.<p>F*** to notate place and padding of the frame number. Sequence start and end frame are separated from filename with ***':'***

'file.%.4F.dpx:0-10' frames 0 to 10 with four digit padding.

FORMATS (out only)

The following are supported by ***--output***

ru Output similar to tweak software 'rvls'. Clip range is embedded in string.
Supports multipart sequences and stepping. Padding is marked either with ***'@'*** or ***'#'.***
'#' means 4 digit padding. ***'@'*** is variable length padding with each ***'@'*** symboling one digit.

'file.10-20#.dpx' frames 10 to 20 with four digit padding
'file.10-30x2,40-50@@.dpx frames 10 to 30 with stepping of 2 and
from 40 to 50 normal stepping. Both clips have padding of 2 digits.

<custom> Any other string passed to format is interpreted as custom format template.
Letters and digits are printed out as such. The following symbols are substituted
with corresponding details of each filesequence:

'%F'	Print out full filesequence string.
'%f'	Same as ***'%F',*** but print out individually each clip in multipart filesequence
'%h'	Head of the filesequence.
'%t'	Tail of the filesequence.
'%s'	Start frame of each clip in filesequence
'%e'	End frame of each clip in filesequence
'%S'	Start frame of each clip in filesequence, with padding
'%E'	End frame of each clip in filesequence, with padding
'%<'	Start timecode of each clip (***--fps*** needs to be set)
'%>'	End timecode of each clip (***--fps*** needs to be set)
'%p'	Padding shown with integers
'%#'	Padding shown with corresponding ***'#'***s
'%@'	Padding shown with corresponding ***'@'***s
'%i'	Increment (stepping) of each clip in sequence
'%n'	Index number (order number) of current sequence in the input
'%L'	Length of each sequence (in frames)
'%-'	Length of each sequence (in timecode, ***--fps*** needs to be set)
'%l'	Length of each clip in sequence (in frames)
'%='	Length of each clip in sequence (in timecode, ***--fps*** needs to be set)

EXAMPLES

Print sequence out with filmlight style

```
rfformat file.[0-10@].dpx --output filmlight
out: file.%.1F.dpx:0-10
```

Take in printf style sequence and return something RV could read

```
rfformat --input printf "file.%04d.dpx 20 30" --output ru
out: file.20-30#.dpx
```

List length of each clip in multipart sequence

```
rfformat  "multipart.[0-10@,40-55@,60-100@].dpx" --fps 25 --output "%f length: %="
out: multipart.[0-10@].dpx length: 00:00:00:11
multipart.[40-55@].dpx length: 00:00:00:16
multipart.[60-100@].dpx length: 00:00:01:16
```

Use in combination with other rftools.
The following shows start and end timecode of each dpx sequence in directory

```
ls | rfpack | rfformat --output "%< %> %f" --fps 25
```

RFBUILD

NAME

rfbuild - Modify the actual files based on the instructions provided by other rftools.

SYNOPSIS

rfbuild [-h] [-ipfsq] [-x] [--ln dir] [--mv dir] [--cp dir] [--rv rv_flags] [--cmd exec args] [filesequence_string ...]

DESCRIPTION

rfbuild performs actions on the files based on instructions piped in from other rftools, usually rfedit. It can be used to link, move (rename), copy filesequence files and it also supports launching selected sequences to Tweak Software RV if rv is installed. It is also possible to run files in filesequence through any custom executable. By default rfbuild runs in 'dry-run' mode when it only displays the commands it is about to run without actually running them. Flag ***-x*** or ***--execute*** is needed to actually perform the actions.

OPTIONS

-h, --help	Display simple help.
-i, --inputdir dir	Use different directory as source root than the current location.
-p, --createdirs dir	Create output directories if they do not exist, similar to mkdir -p.
-f, --force	Force overwrite if there are conflicting files in copy, move or link operation. Not recommended.
-s, --skip	Skip warnings about missing source files and do actions on those that are found.
-q, --quiet	Supress messages to standard output.
-x, --execute	Actually perform the actions, otherwise only shows the commands as a 'dry-run' of what would happen.

ACTIONS

Actions are performed from source files to target files. This means that the filesequence descriptions have been manipulated by rfedit or that input and output directory are different.
Otherwise rfbuild will exit with 'nothing to do'

-l, --ln output dir	Create unix soft links from source files to target files.
-m, --mv output dir	Move files based on input instructions or do simple move from input dir to output dir.
-c, --cp output dir	Copy files based on input instructios or do simple copy from input dir to output dir.
-r, --rv rv_flags	If Tweak Software RV is installed and can be launched from commandline this flag will launch source sequences to RV. Alternatively RV_EXECUTABLE_PATH environment variable can point to rv executable. Arguments following ***--rv*** are flags passed down to RV executable. Use ***'--'*** for no flags. If you have several flags to RV quote them ***"-fullscreen -sRGB".***
--cmd exec arguments	***--cmd*** can be used to execute custom program or shellscript for each file. exec is the path to executable. Arguments can be any flags passed to executable or wildcards ***%original*** and ***%modified*** which will be replaced by each filename in sequence.

EXAMPLES

Create combined sequence by softlinking each frame of all sequences in current subdirectories.
The combined sequence called ***combined.####.dpx"*** will be created inside directory ***'new_seq'*** which will be created during the build.

> find . | rfpack | rfedit --reconstruct "combined.[0-1000@@@@].dpx" | rfbuild --ln ./new_seq -x -p

Move (rename) 0-based sequence to 1-based in the same directory.

> ls | rfpack | rfedit --start 1 | rfbuild --mv . -x

Copy files missing from local 'my_sequence' directory from remote server

> ls my_sequence | rfpack -i | rfbuild --inputdir /mnt/server/remoteseq --cp my_sequence/ -x

Find all dpx sequences in all subdirectories of current directory and load them to rv.

> find . -type f -name "*.dpx" | rfpack | rfbuild --rv "-fullscreen"

Generate seqeunce of 25 empty text files using command ***'touch'***

> rfbuild "empty.[1-25@@@@].txt" --cmd touch "%original" -x