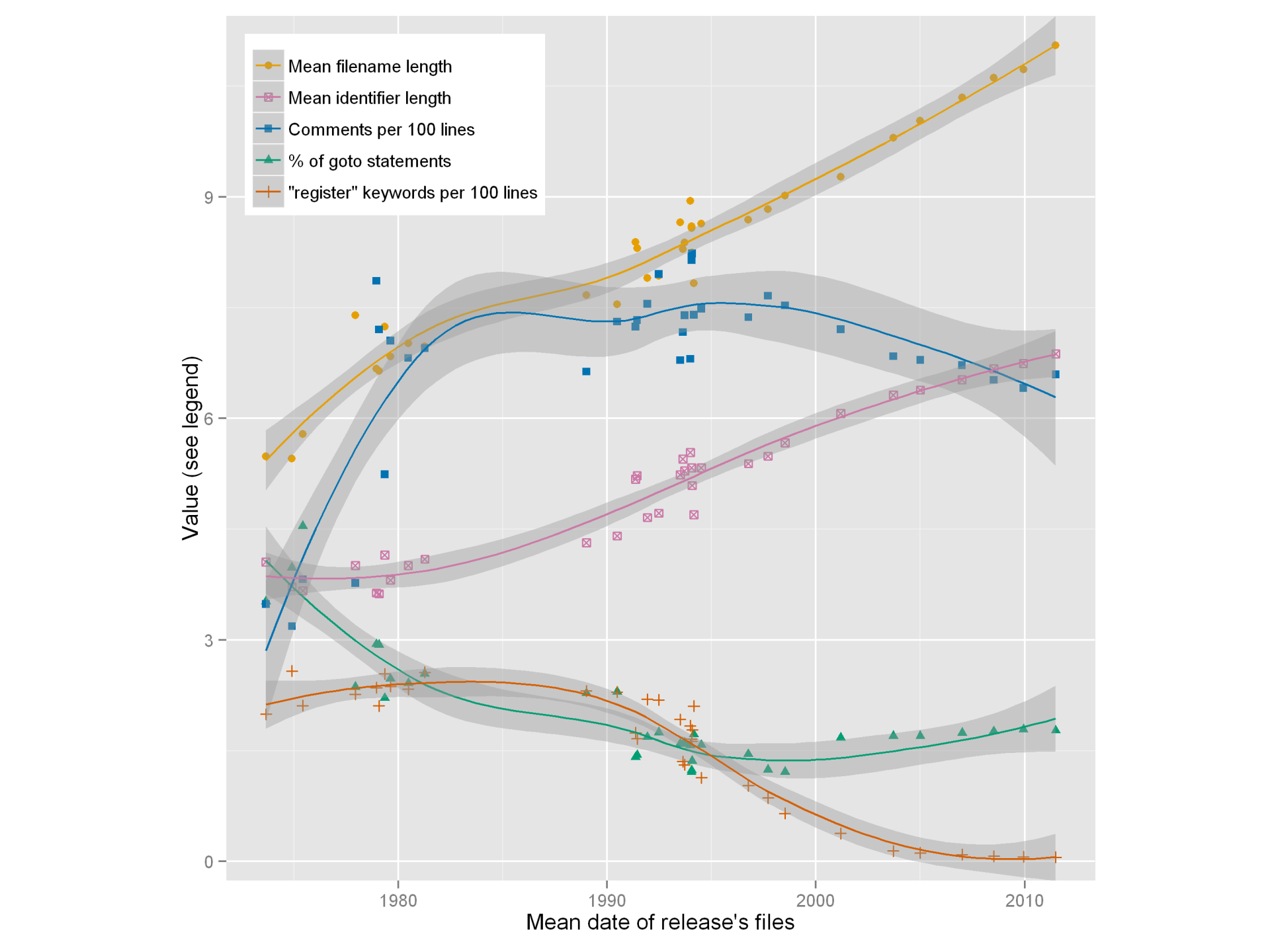
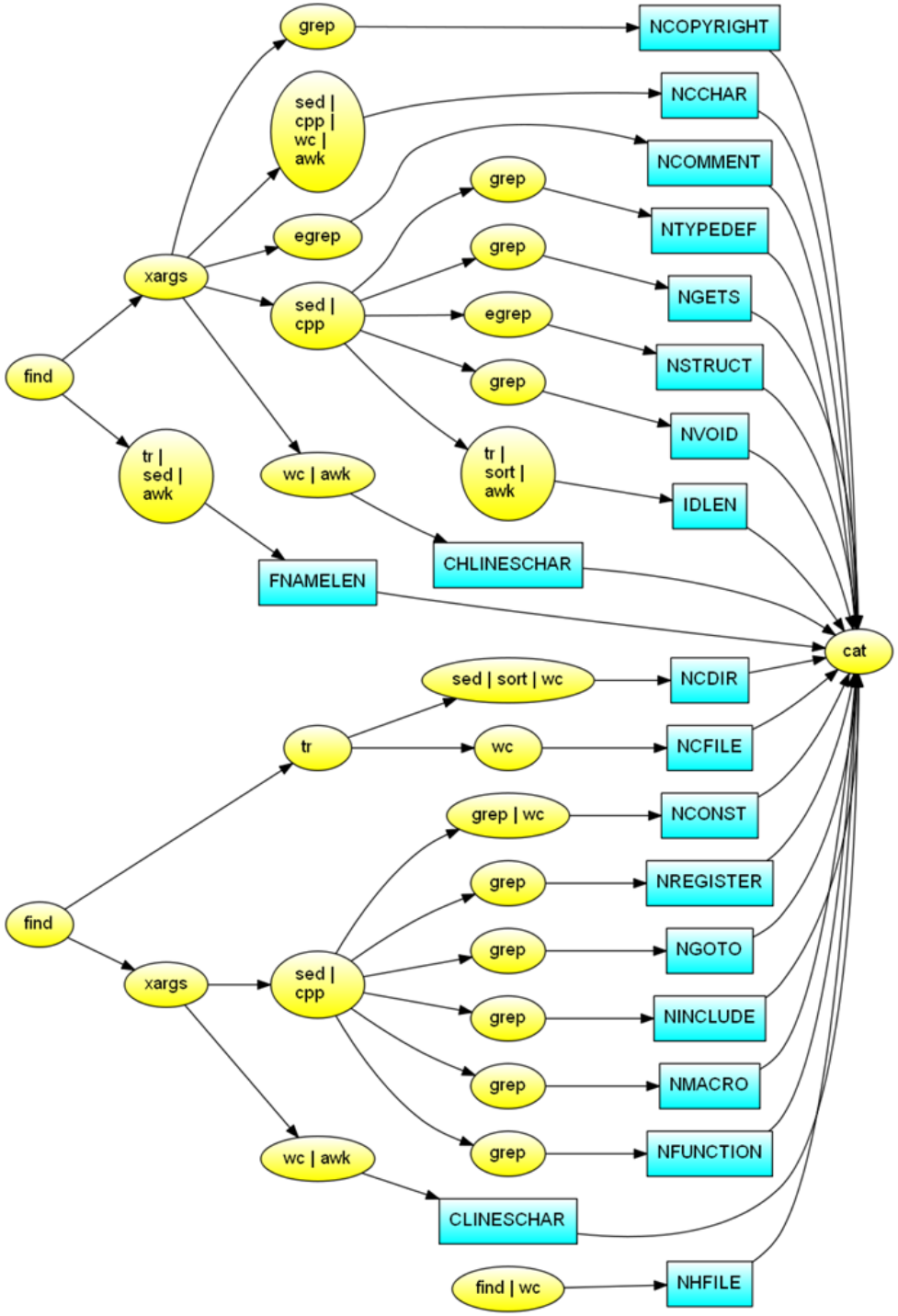
From pipelines to graphs: Escape the tyranny of the shell's linear pipelines with **dgsh**

Diomidis Spinellis, Marios Fragkoulis
Department of Management Science and Technology
Athens University of Economics and Business

www.spinellis.gr
{dds,mfg}@aueb.gr
@CoolSWEng

```
curl http://example.com/doc.txt |
tr -C a-zA-Z '\n' |
tr A-Z a-z |
sort |
uniq |
comm -23 - /usr/share/dict/words
```


Mean filename length
Mean identifier length
Comments per 100 lines
% of goto statements
"register" keywords per 100 lines
Value (see legend)
0
3
6
9
1980
1990
2000
2010
Mean date of release's files


grep
NCOPYRIGHT
sed | cpp | wc | awk
NCCHAR
NCOMMENT
egrep
grep
NTYPEDEF
xargs
grep
NGETS
sed | cpp
egrep
NSTRUCT
find
grep
NVOID
tr | sed | awk
wc | awk
tr | sort | awk
IDLEN
CHLINESCHAR
FNAMELEN
cat
sed | sort | wc
NCDIR
tr
wc
NCFILE
grep | wc
NCONST
find
grep
NREGISTER
grep
NGOTO
xargs
sed | cpp
grep
NINCLUDE
grep
NMACRO
grep
NFUNCTION
wc | awk
CLINESCHAR
find | wc
NHFILE

Alternatives

- Run each pipeline separately
- Temporary files
- bash >(*process*) syntax:

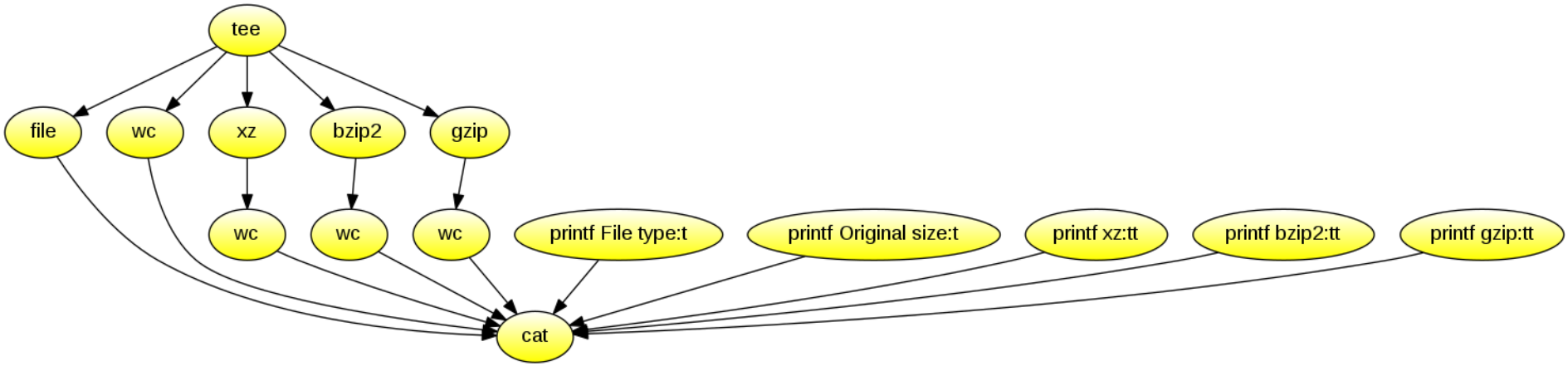
```
find . -name \*.c …|
tee >(
    tee >(wc -l >NCFILE)
    tee >(sed … | wc -l >NCDIR)
) >(
    xargs cat | tee >(…
```

dgsh

Compression comparison

```
$ compress-compare http://www.gutenberg.org/files/28054/28054-0.txt
File type: /dev/stdin: UTF-8 Unicode text
Original size: 2044191
xz:              581816
bzip2:           539868
gzip:            745027
```

Compression comparison



Compression comparison

```
curl -s "$1" |
tee |
{{
        printf 'File type:\t' &
        file - &

        printf 'Original size:\t' &
        wc -c &

        printf 'xz:\t\t' &
        xz -c | wc -c &

        printf 'bzip2:\t\t' &
        bzip2 -c | wc -c &

        printf 'gzip:\t\t' &
        gzip -c | wc -c &
}} |
cat
```

IPC Mechanisms

- Multipipe block
  {{
  command1 &
  command2 &
  …
  }}

- Unix commands with multiple I/O channels
- Stored values

dgsh-aware programs

Tool	Input Channels	Output Channels
tee	1	0–N
cat	0–N	1
comm	0–2	0–3
diff	0–2	0–1
grep	0–2	0–1
join	0–2	0–1
paste	0–N	1
perm	1–N	1–N
sort	0–N	0–1
dgsh-readval	0	1
dgsh-writeval	0–N	0–1
dgsh-wrap	1	0

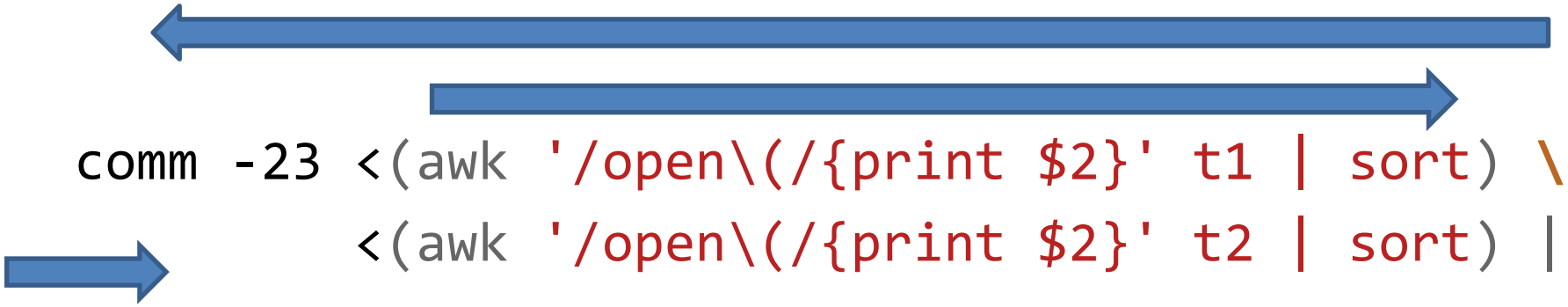
API

```
#include "dgsh.h"

int
dgsh_negotiate(const char *tool_name,
 int *n_input_fds,
 int *n_output_fds,
 int **input_fds,
 int **output_fds);
```

Backward compatibility

- dgsh-wrap
- Protocol wrapper for existing commands
- Can state command's I/O requirements
  - e.g. echo, ls, find take no input
- <- and >- for specifying I/O channels as file arguments

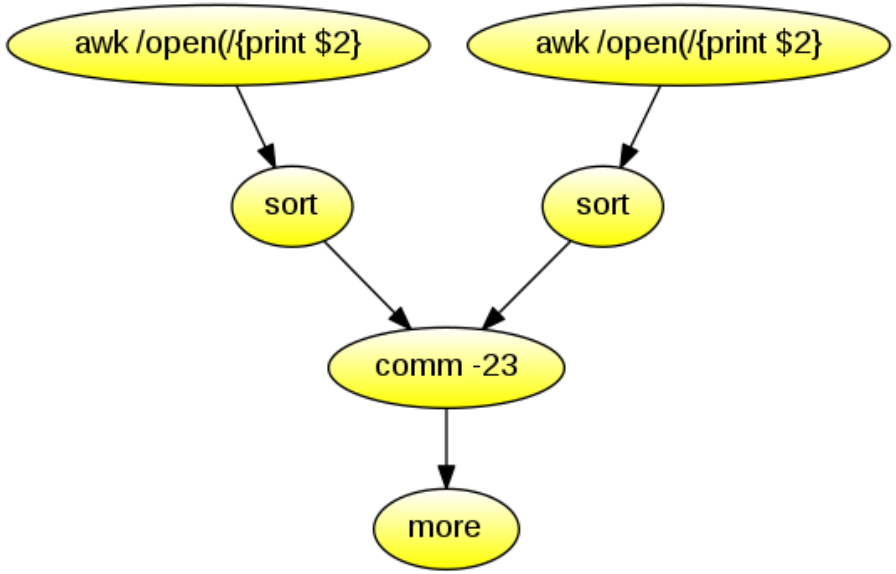
Debug configuration difference



```
comm -23 <(awk '/open\(/{print $2}' t1 | sort) \
         <(awk '/open\(/{print $2}' t2 | sort) |
more
```

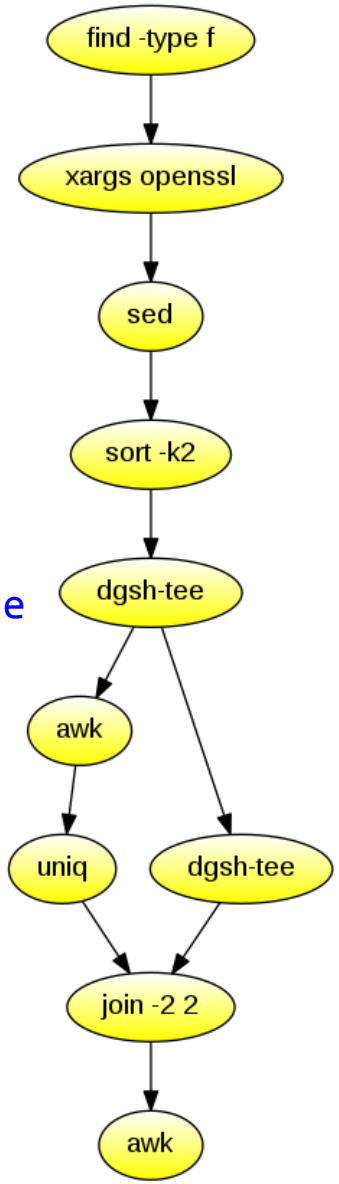
Debug configuration difference

```
{{
  awk '/open\(/{print $2}' t1 | sort &
  awk '/open\(/{print $2}' t2 | sort &
}} |
comm -23 |
more
```



Duplicate files

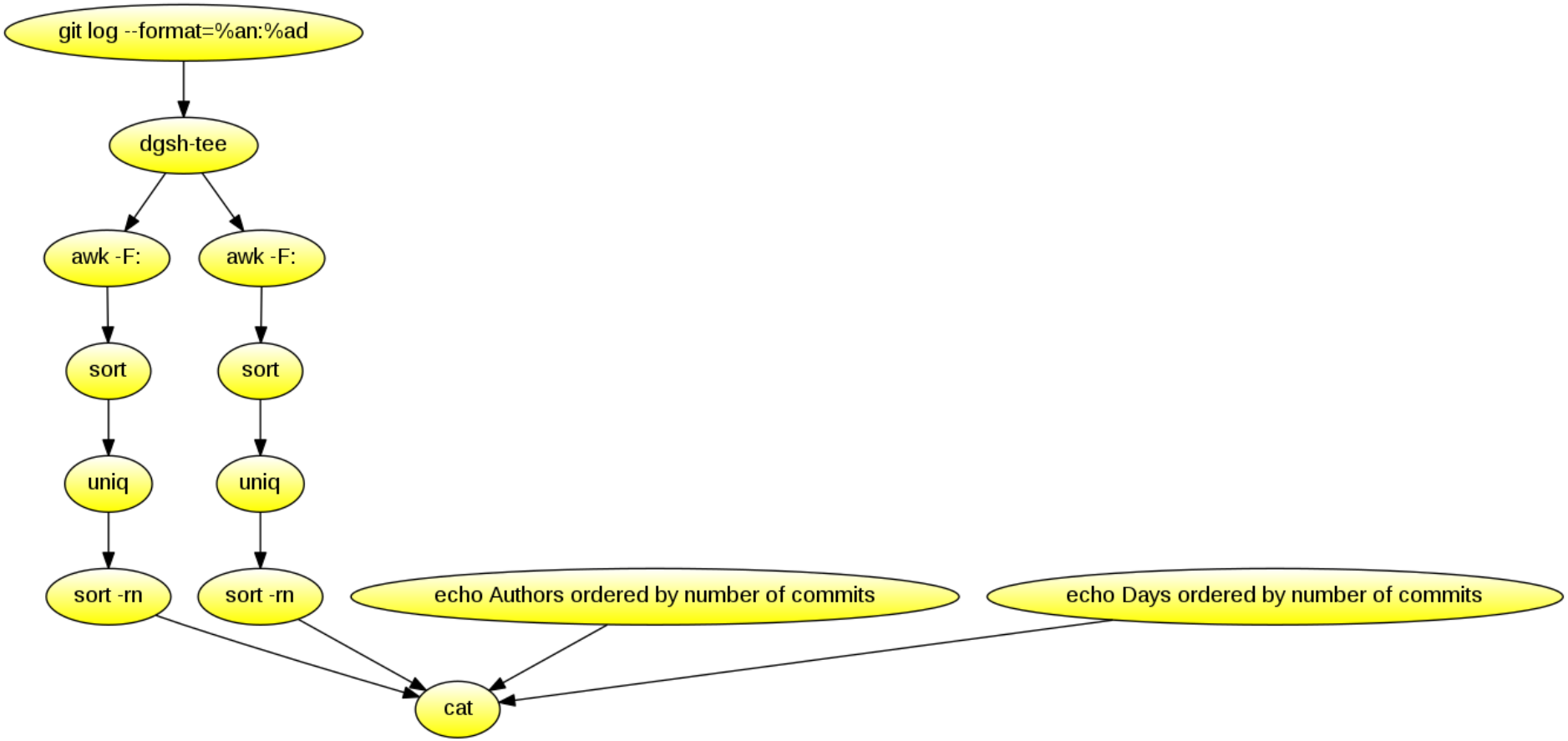
```
#!/usr/bin/env dgsh

# Create list of files
find "$@" -type f |
# Produce lines of the form
# MD5(filename)= 811bfd4b5974f39e986ddc037e1899e7
xargs openssl md5 |
# Convert each line into a "filename md5sum" pair
sed 's/^MD5(//;s/)= / /' |
# Sort by MD5 sum
sort -k2 |
tee |
{{
        # Print an MD5 sum for each file that appears more than once
        awk '{print $2}' | uniq -d &
        cat &
}} |
# Join the repeated MD5 sums with the corresponding file names
# Join expects two inputs, second will come from scatter
join -2 2 |
# Output same files on a single line
awk '
BEGIN {ORS=""}
$1 != prev && prev {print "\n"}
END {if (prev) print "\n"}
{if (prev) print " "; prev = $1; print $2}'
```



Duplicate files

```
$ duplicate-files.sh /lib
/lib/xtables/libxt_conntrack.so /lib/xtables/libxt_state.so
/lib/xtables/libxt_CT.so /lib/xtables/libxt_NOTRACK.so
```

Git commit metrics

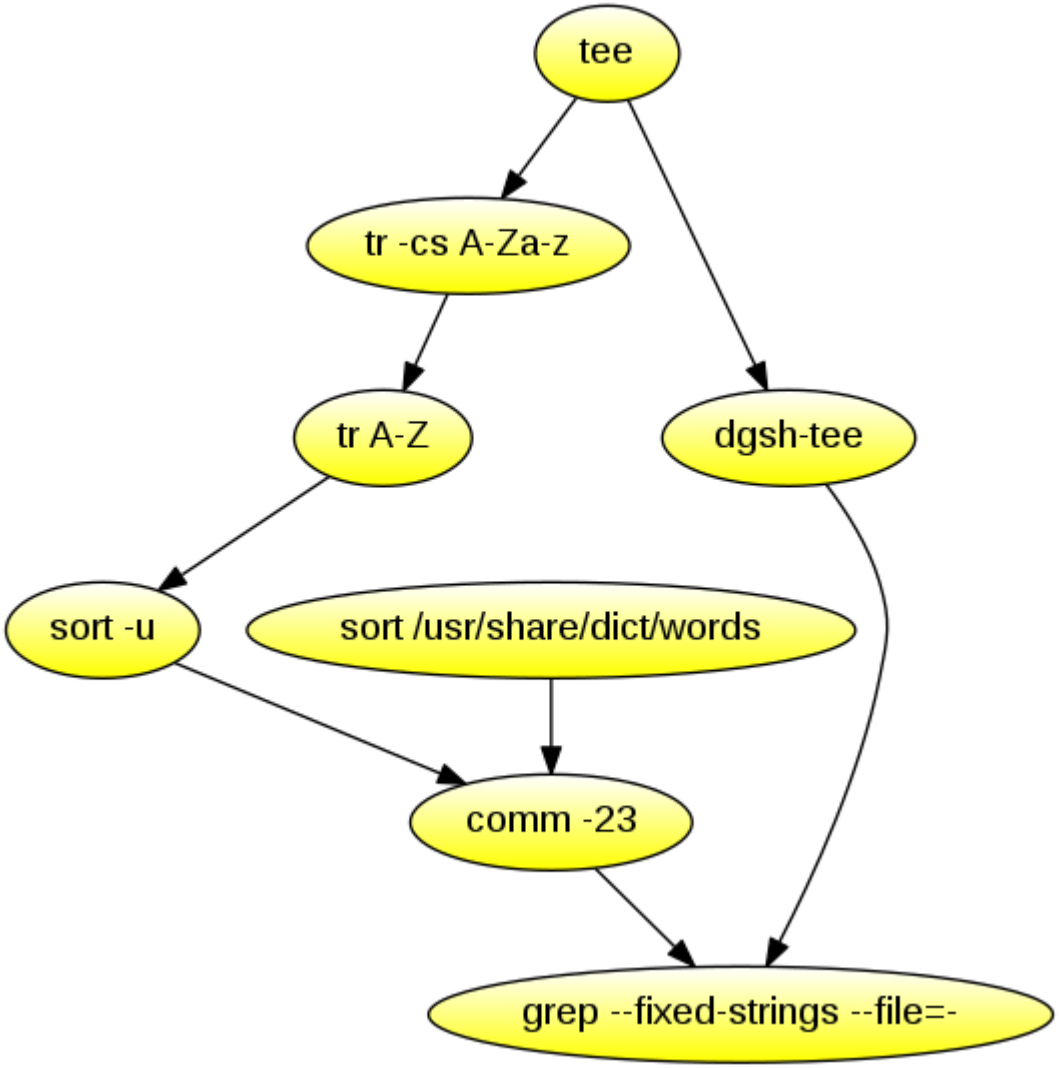


Git commit metrics

```
# Order by frequency
forder()
{
        sort |
        uniq -c |
        sort -rn
}

git log --format="%an:%ad" --date=default "$@" |
tee |
{{
    echo "Authors ordered by number of commits" &
    awk -F: '{print $1}' | forder &

    echo "Days ordered by number of commits" &
    awk -F: '{print substr($2, 1, 3)}' | forder &
}} |
cat
```

```
Authors ordered by number of
commits
  20140 Linus Torvalds
   8445 David S. Miller
   7693 Andrew Morton
   5156 Greg Kroah-Hartman
   5118 Mark Brown
   4723 Russell King
   4586 Takashi Iwai
   4388 Al Viro
   4220 Ingo Molnar
   3277 Tejun Heo
   3245 H Hartley Sweeten
   ...
Days ordered by number of commits
  88683 Tue
  86987 Wed
  85407 Mon
  83660 Thu
  74814 Fri
  37868 Sun
  34562 Sat
```

Highlight misspelled words



Highlight misspelled words

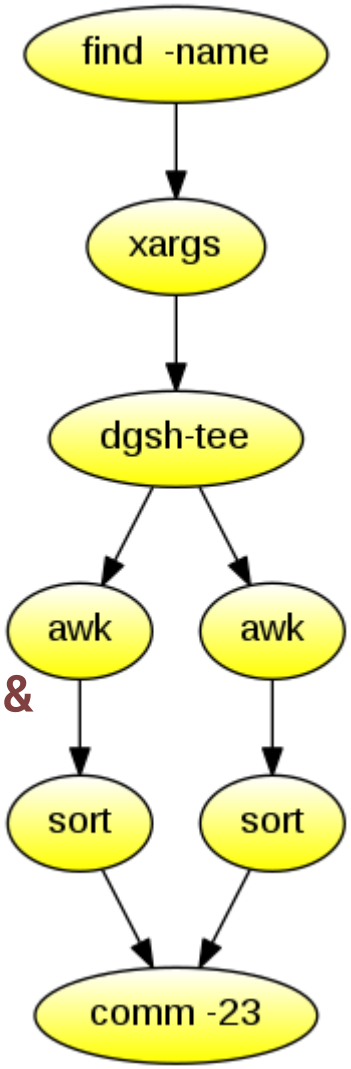
```
tee |
{{      # Find errors
        {{
                # Obtain list of words
                tr -cs A-Za-z \\n |
                tr A-Z a-z |
                sort -u &

                # Ensure dictionary is compatibly sorted
                sort /usr/share/dict/words &
        }} |
        comm -23 &

        # Pass through text
        cat &
}} |
grep --fixed-strings --file=- --ignore-case --color \
  --word-regex --context=2
```

Highlight misspelled words

```
$ curl -s http://www.gutenberg.org/files/74/74.txt |
> sed -n '/^"Well, that sounds like/,/^"Have you?/p' |
> example/spell-highlight.sh
"Well, that sounds like a good way; but that ain't the way Bob Tanner
done."

"No, sir, you can bet he didn't, becuz he's the wartiest boy in this
town; and he wouldn't have a wart on him if he'd knowed how to work
spunk-water. I've took off thousands of warts off of my hands that way,
Huck. I play with frogs so much that I've always got considerable many
warts. Sometimes I take 'em off with a bean."

"Yes, bean's good. I've done that."

"Have you? What's your way?"
$
```

C/C++ symbols that should be static

```
# Find object files
find "$1" -name \*.o |

# Print defined symbols
xargs nm |

tee |
{{
  # List all defined (exported) symbols
  awk 'NF == 3 && $2 ~ /[A-Z]/ {print $3}' | sort &

  # List all undefined (imported) symbols
  awk '$1 == "U" {print $2}' | sort &
}} |

# Print exports that are not imported
comm -23
```

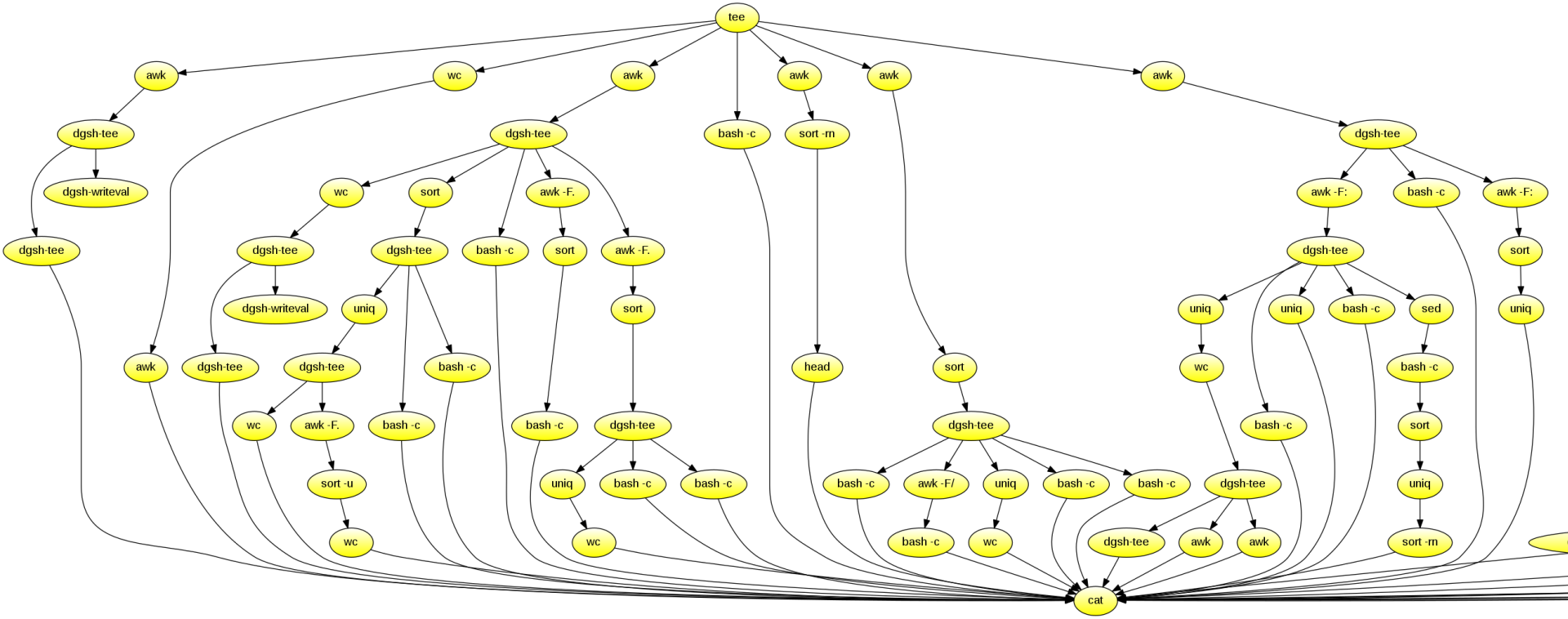


C/C++ symbols that should be static

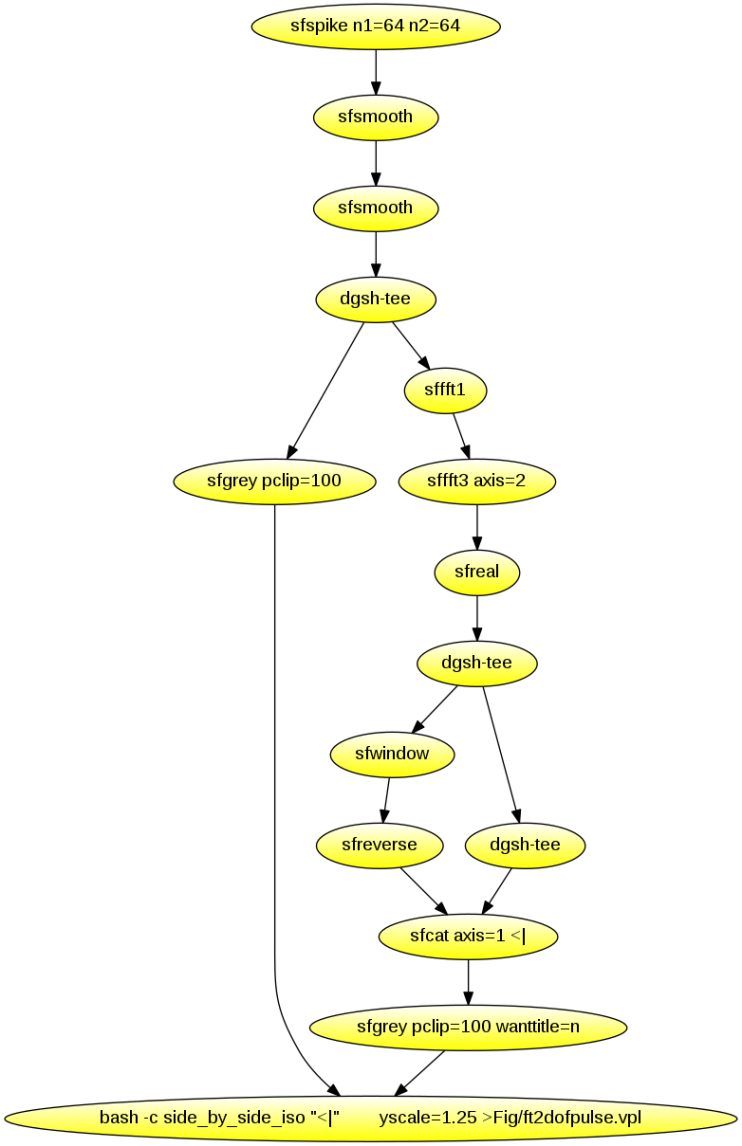
```
$ static-functions.sh bash
add_documentation
add_or_supercede_exported_var
alias_doc
alias_expand_all
alias_expand_word
alloc_history_entry
alphabetic
ansicstr
ansic_wshouldquote
arith_doc
arith_for_doc
array_add
array_assign_list
array_create_element
array_dequote
array_dequote_escapes
```

The list goes on …

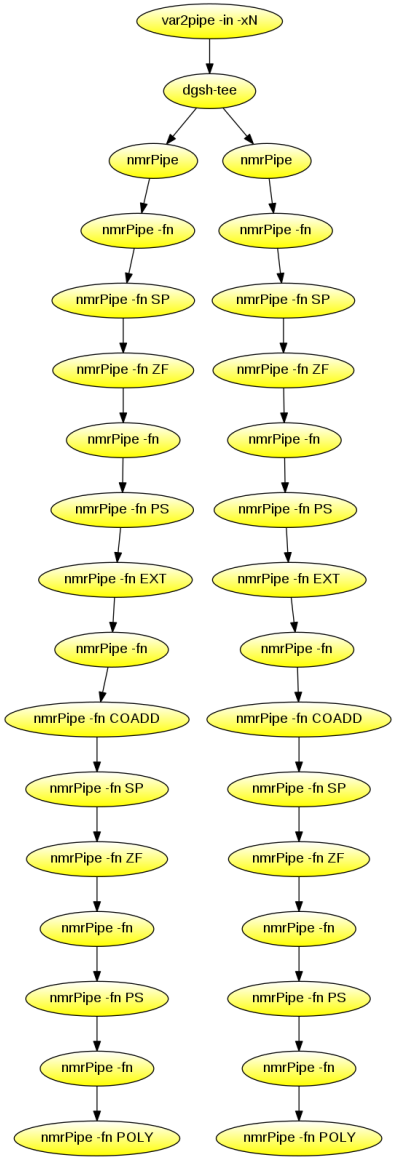
Web log reporting



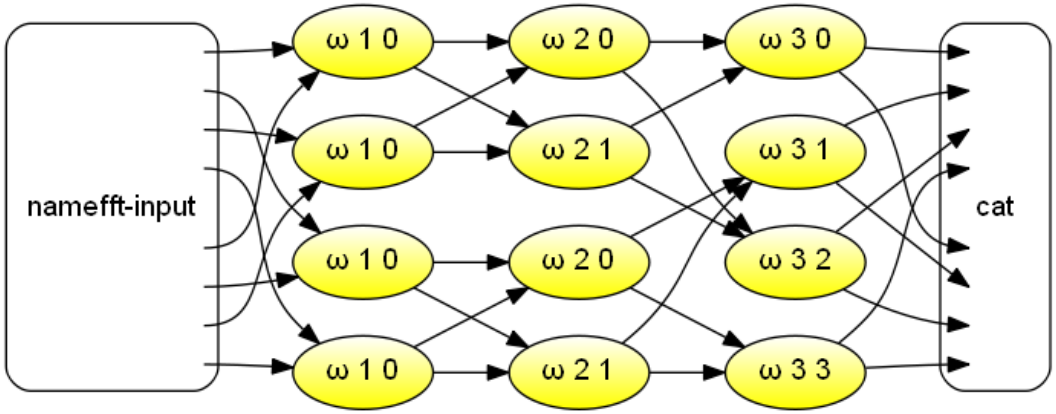
Waves: 2D Fourier transforms

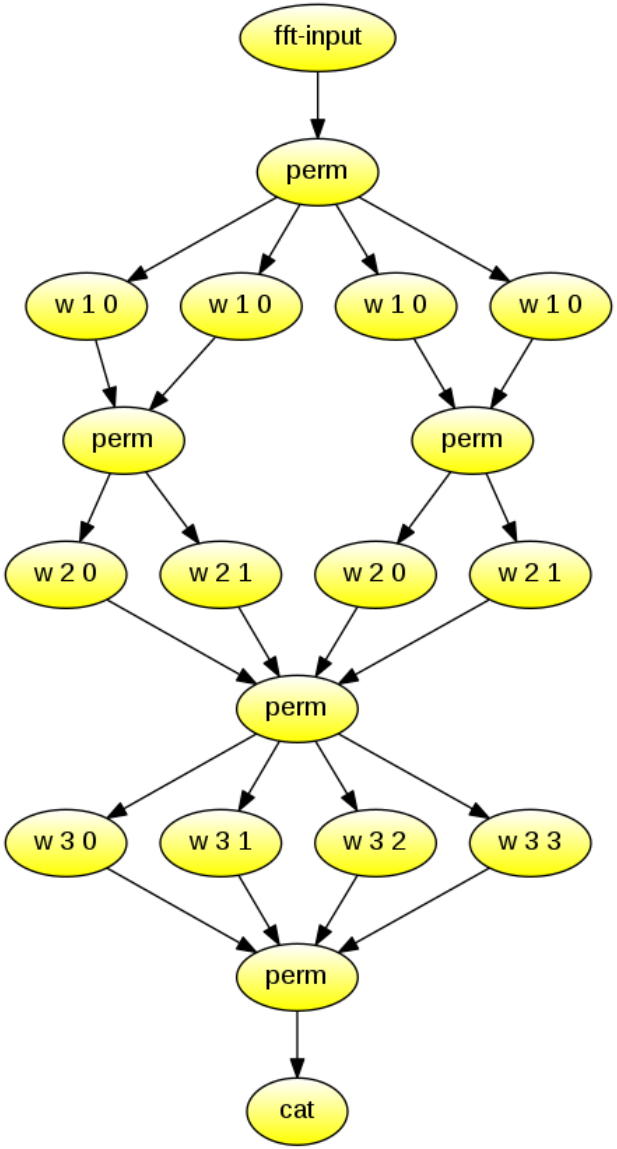


NMR Processing

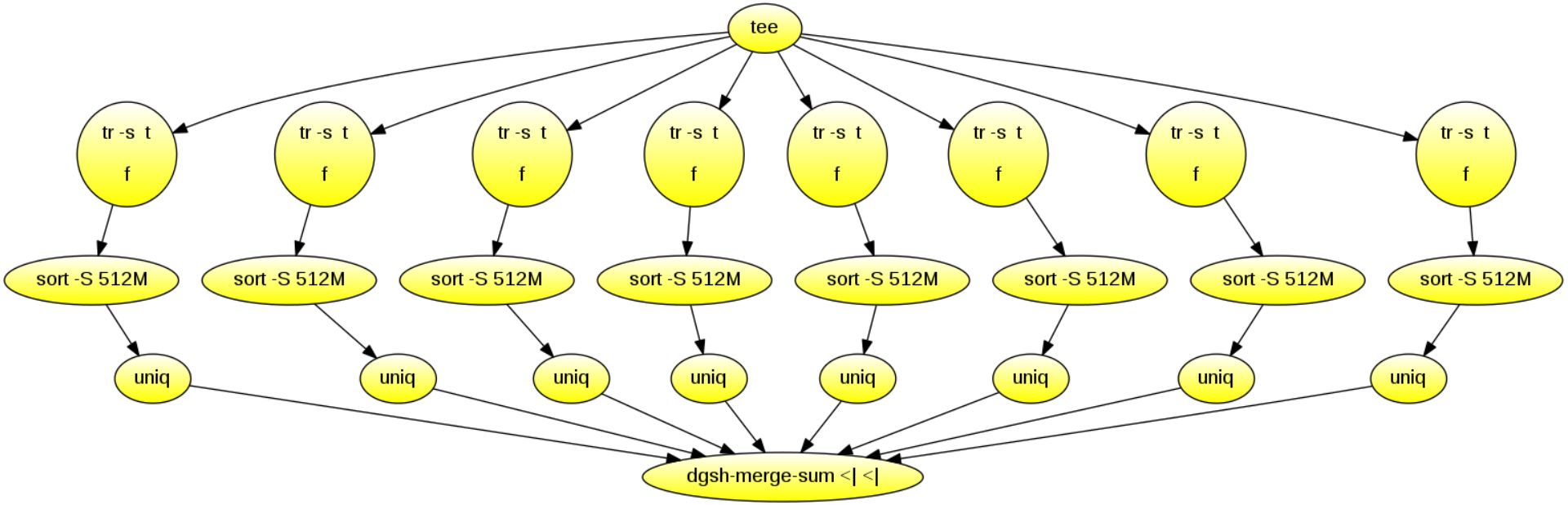


Parallel FFT computation

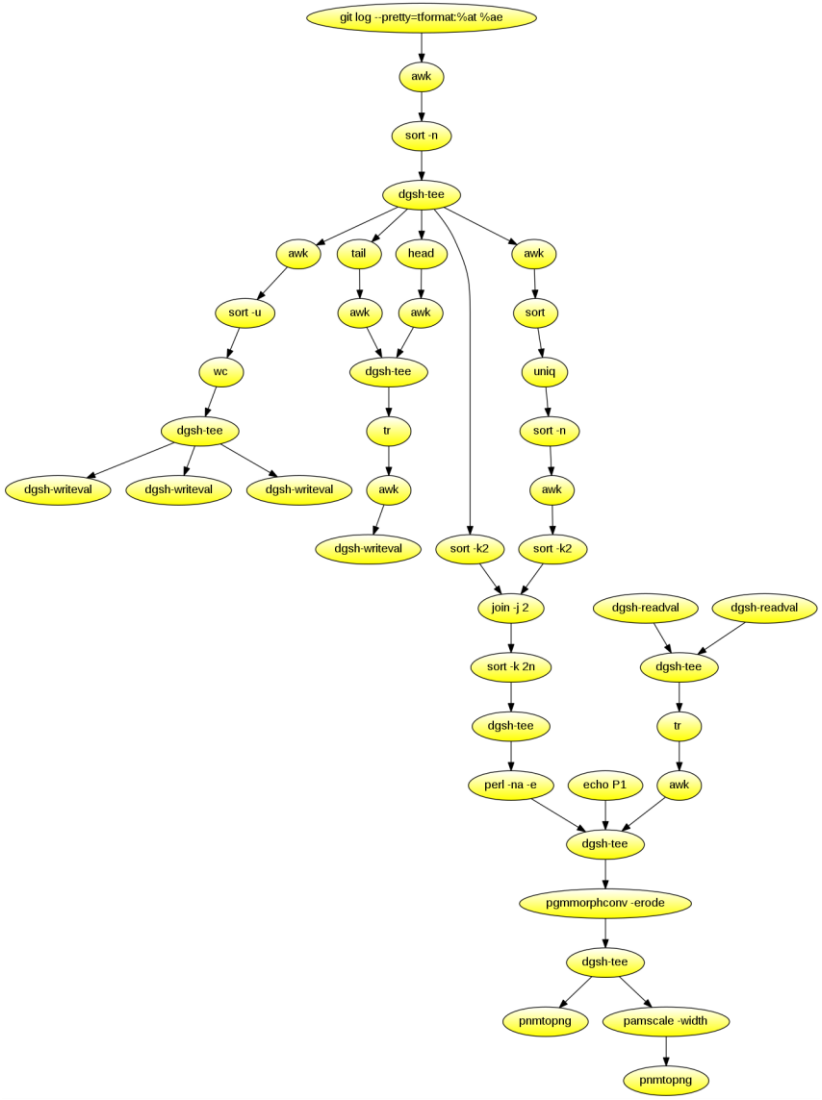




Parallel word count



Draw Git committer activity over time



// TODO

github.com/dspinellis/dgsh


18v


MARCO & RAF
GUERREIRO
ZOUKA
AUTO BLOK
Auto
Prestige
Apidex
PF
Motorsport
DOTZ
KW
suspensions
RF-BAT
GarageACC
CARROSSERIE
DEMIE LUNE
OSPC
EXPERT
Auto
ALIENTECH
Allianz
JOSE ASSUNCAO
KW
suspensions
MOTUL
FEDERAL

FOSDEM Treasure Hunt

dgsh — directed graph shell

The directed graph shell, *dgsh* (pronounced /dæg[/ — *dagsh*), provides an expressive way to construct sophisticated and efficient big data set and stream processing pipelines using existing Unix tools as well as custom-built components. It is a Unix-style shell (based on *bash*) allowing the specification of pipelines with non-linear non-uniform operations. These form a directed acyclic process graph, which is typically executed by multiple processor cores, thus increasing the operation's processing throughput.

If you want to get a feeling on how *dgsh* works in practice, skip right down to the examples section.

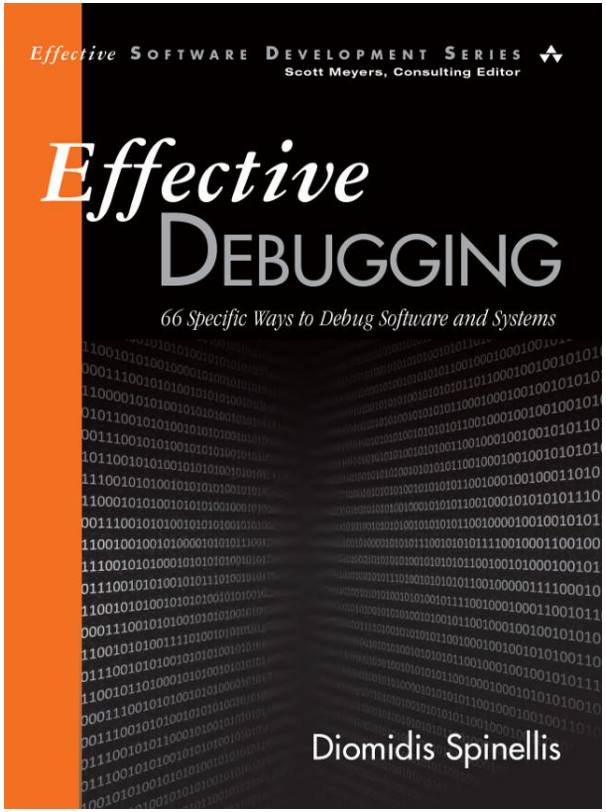
Inter-process communication

Dgsh provides two new ways for expressing inter-process communication.

Multipipes are expressed as usual Unix pipelines, but can connect commands with more than one output or input channel. As an example, the `comm` command supplied with *dgsh* expects two input channels and produces on its output three output channels: the lines appearing only in first (sorted) channel, the lines appearing only in the second channel, and the lines appearing in both. Connecting the output of the `comm` command to the `cat` command supplied with *dgsh* will make the three outputs appear in sequence, while connecting it to the `paste` command supplied with *dgsh* will make the output appear in its customary format.

Multipipe blocks {{ ... }} a) send (multiple) input streams received on their input side to the asynchronously-running processes that reside within the block, and, b) pass the output produced by the processes within the block as (multiple) streams on their output side. Multipipe blocks typically receive input from more than one channel and produce more than one output channel. For example, a multipipe block that runs `md5sum` and `wc -c` receives two inputs and produces two outputs: the MD5 hash of its input and the input's size. Data to multipipe blocks are typically provided with a *dgsh*-aware version of `tee` and collected by *dgsh*-aware versions of programs such as `cat` and `paste`.

Stored values offer a convenient way for communicating computed values between arbitrary processes on the graph. They allow the storage of a data stream's last record into a named buffer. This record can be later retrieved asynchronously by one or more readers. Data in a stored value can be piped into a process or out of it, or it can be read using the shell's command output substitution syntax. Stored values are implemented internally through Unix-domain sockets, a background-running store program, `dgsh-writeval`, and a reader program, `dgsh-readval`. The behavior of a stored value's IO can be modified by adding flags to `dgsh-writeval` and `dgsh-readval`.



Thank you!

github.com/dspinellis/dgsh

{dds,mfg}@aueb.gr

www.spinellis.gr

@CoolSWEng

Looking for …

PhD Candidate or Postdoctoral Researcher in Applied Software Engineering

- Mine software repositories to improve IDE recommendations
- Prerequisites
  - MSc || PhD && strong technical background
- Funded until end of 2019
- Partners: AUEB, Eclipse Foundation, CWI, Univ York, …
- Info: https://euraxess.ec.europa.eu/jobs/175366

Image Credits

Pipeline: Diego Delso CC BY-SA 4.0
Copper pipe fittings: Torsten Bätge CC BY-SA 3.0
Sliced bread: Matt Burns CC BY-SA 2.0
Package: Andrew Bowden CC BY-SA 2.0
Power tool: Matt DeTurck CC BY-NC-ND 2.0
Enhanced car: www.twin-loc.fr CC BY 2.0