THE PARTS-OF-FILES FILE SYSTEM

YOANN PADIOLEAU and OLIVIER RIDOUX

IRISA / University of Rennes 1

{padiolea,ridoux}@irisa.fr

http://www.irisa.fr/LIS

CFSE, 2003

Plan

- Motivations (several points of view on file contents)
- An example (views of a program file)
- Specification (cd = ?, ls = ?, emacs = ?)
- Implementation (data structures and algorithm)
- Evaluation (time and space)
- Related works (file systems and integrated development environments)
- Conclusion and further works (improve PofFS, validate usage)

Motivation (1)

Abstraction from physical lay-out

- directory trees and flat files reproduce physical constraints

  1 object $\Longrightarrow$ 1 place $\Longrightarrow$ 1 classification $\Longrightarrow$ 1 point of view

- applications & users want several points of view

  end-user applications: music files (genre, author, year. . .)

  professional applications: software (code, spec, doc,. . .)

- we have proposed Logical File System (LISFS) as a substitute to directory trees [USENIX 2003, Annual Technical Conference]

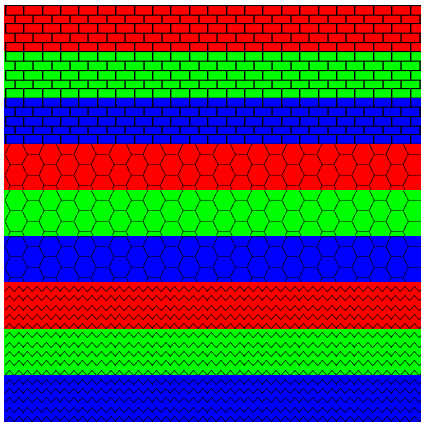
  navigation and querying among files

- the missing link: navigation and querying inside files

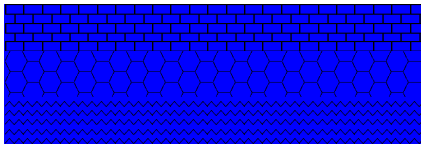
Motivation (2)

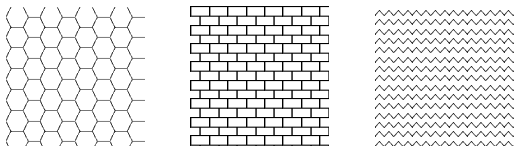
Cross-cutting points of view

- programs: {functions, variables, types}
  $\times$ {comments, declarations, specifications}
- books: {abstract, sections, index}
  $\times$ {topics, keywords, citations}



- `% cd blue`
  `% ls`
  `the_blue_parts`
  `honeycomb/ brick/ wave/`
  `% emacs brick`





Motivation (3)

query, navigation, editing of points of view on file contents

- better understanding
- easy focus on one task
- coherent change

An example

views of a program file

```
[1] % cat -n foo.c
1 int f(int x) {
2 int y;
3 assert(x > 1);
4 y = x;
5 fprintf(stderr, "x = %d", x);
6 return y * 2
7 }
8 int f2(int z) {
9 return z * 4
10 }
```

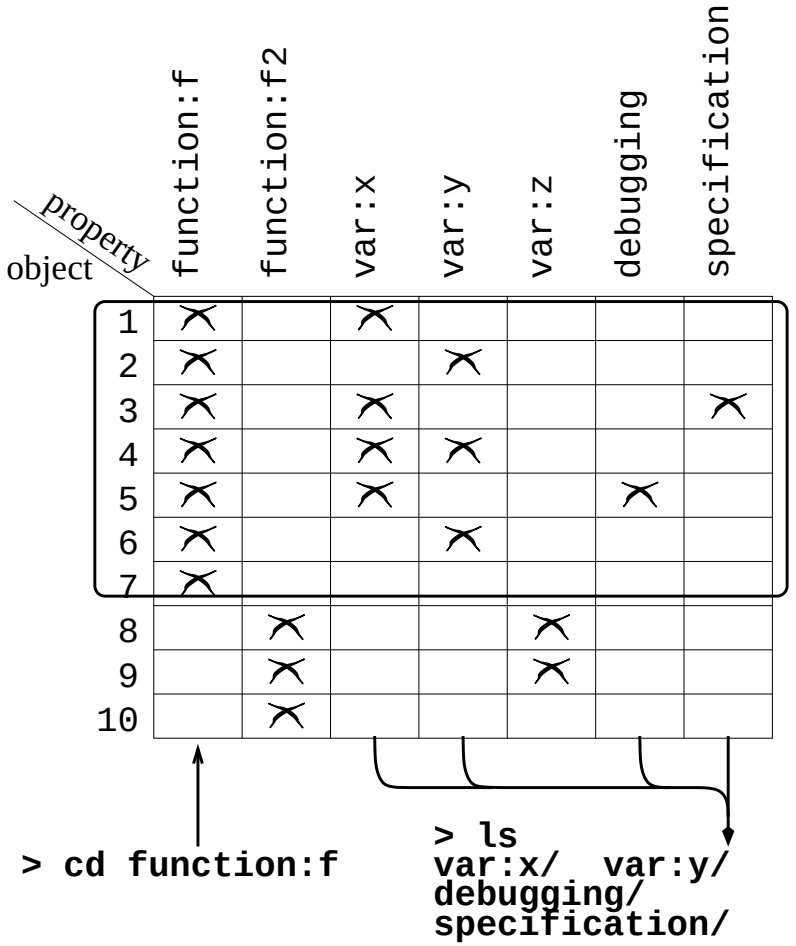
indexing

```
[2] % poffsmount
        foo.c /poffs
        --transducers=
        c_transducer,
        /home/pad/my_transducer
```

property / object	function:f	function:f2	var:x	var:y	var:z	debugging	specification
1	X		X				
2	X			X			
3	X		X				X
4	X		X	X			
5	X		X			X	
6	X			X			
7	X						
8		X			X		
9		X			X		
10		X					

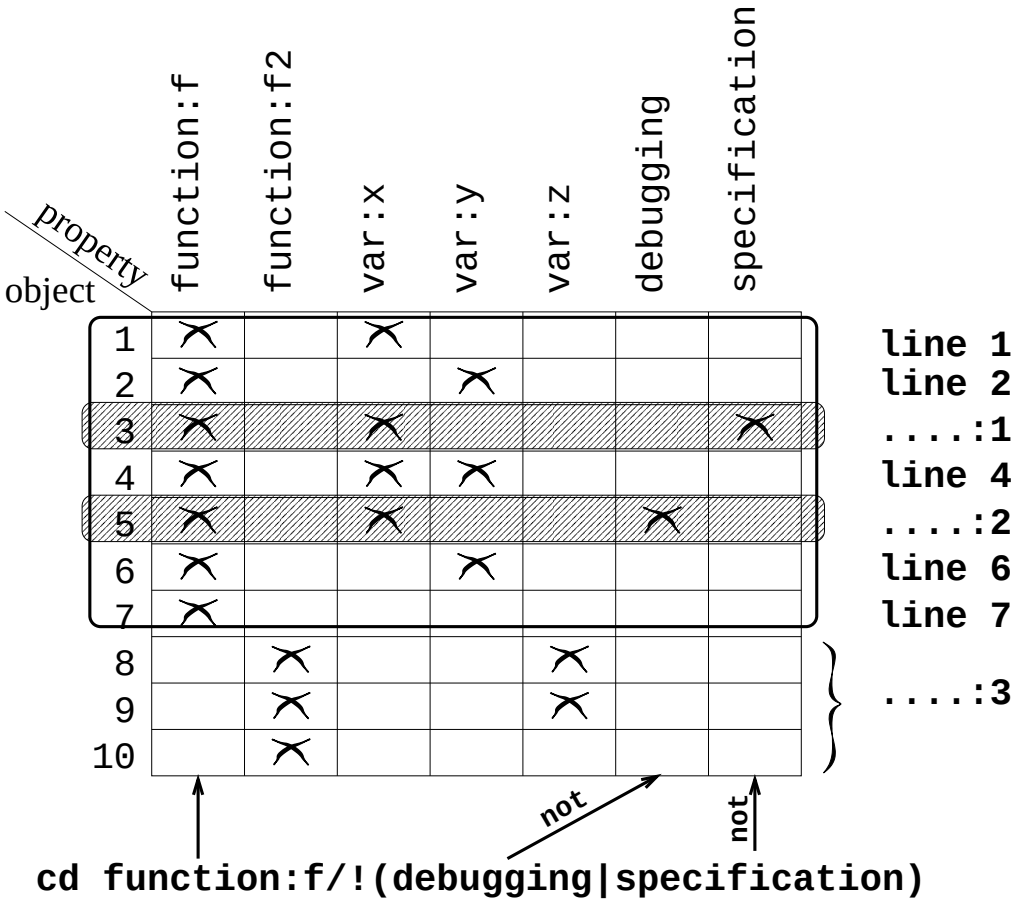
navigating

```
[3] % cd /poffs
[4] % ls
foo.c
debugging/
specification/
function:f/
function:f2/
var:x/ var:y/ var:z/
[5] % cd function:f
[6] % ls
foo.c
debugging/
specification/
var:x/ var:y/
```



querying

```
[7] % cd
!(debugging|specification)
[8] % ls
foo.c var:x/ var:y/
[9] % cat foo.c
int f(int x) {
int y;
.................:1
y = x;
.................:2
return y * 2
}
.................:3
```



updating

```
[10] % cat foo.c | sed -e s/y/z > foo.c
[11] % ls
foo.c var:x/ var:z/
[12] % cd /poffs
[13] % cat foo.c
int f(int x) {
int z;
assert(x > 1);
z = x;
fprintf(stderr, "x = %d", x);
return z * 2
}
inf f2(int z) {
return z * 4
}
```

Important notions (1)

PofFS indexing

- a logic — $A \models B$

  deduction rules

  and axioms

  ($A \wedge B \models A$

  and *else* $\models$ *cond*)

- information — an attachment d (description) of a logic formula to every part $o \in \mathcal{O}$ (objects) of a file

  ($d($`y = x;`$) =$ *var:x* $\wedge$ *var:y* $\wedge$ *function:f*)

- transducers — user-defined, they compute d (PofFS plug-ins)

  (java $\rightarrow$ class:Button, . . .)

transducers are finite-state machines

Important notions (2)

querying PofFS

paths are formulas

- extension — given a path p, the set all parts-of-files that satisfy this property

$$ext(p) = \{o \in \mathcal{O} \mid d(o) \models p\}$$

extensions are presented as physical views

paths denote directories that denote extensions

(working directory $\equiv$ working query $\equiv$ working view)

Important notions (3)

navigating PofFS

- subdirectories — given a directory O, every directory O' such that $O' \subsetneq O$

$$Dirs(p) = max_{\models}\{p' \in \mathcal{L} \mid \emptyset \subsetneq ext(p \wedge p') \subsetneq ext(p)\}$$

(p' refines p)

only largest subdirectories are relevant to navigation

(most relevant hints)

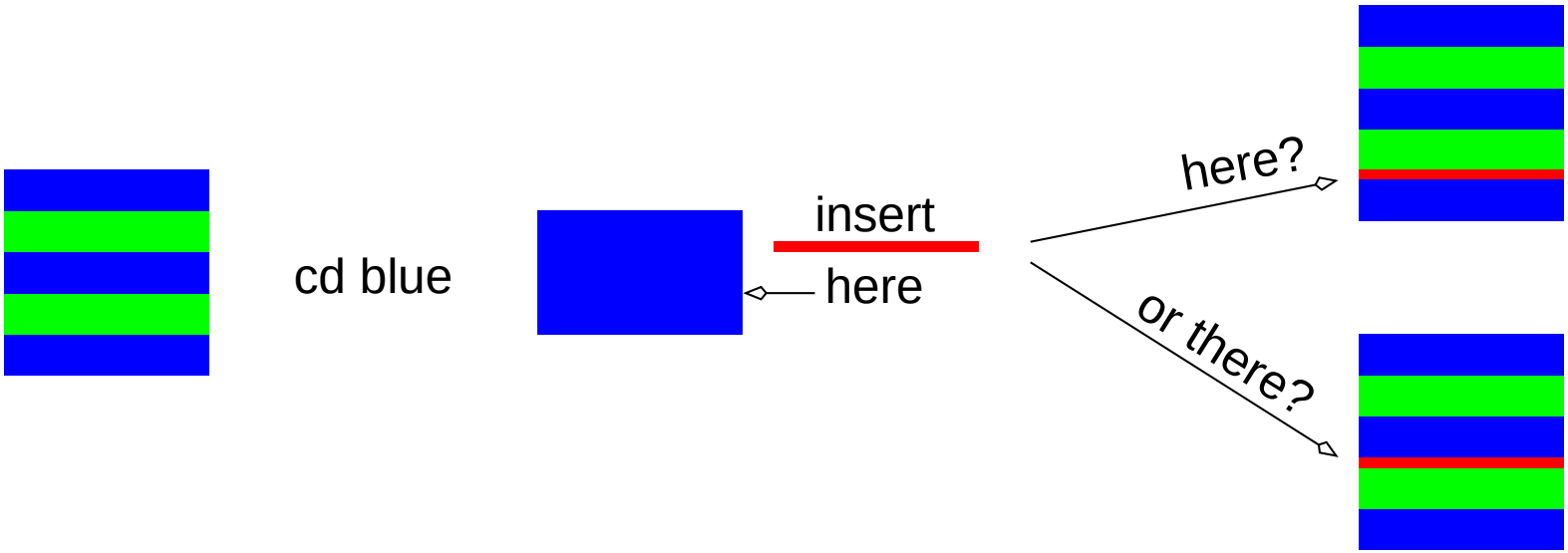
- to be a file of a directory — the physical view formed after the extension of a path p

$$\{ext(p)\} \bigcup Dirs(p) \quad \approx \quad \text{ls } p$$

Important notions (4)

updating PofFS

- re-indexing — updating a physical view $\longrightarrow$ updating the object$\times$property matrix,
- marks of missing parts — . . . . . :X



to know where to insert new parts

Implementation

to implement the specification at a reasonable cost

basic principles

- to call $\models$ is costly

  represent relation d as a table and inverted table on disk

  (faster querying)

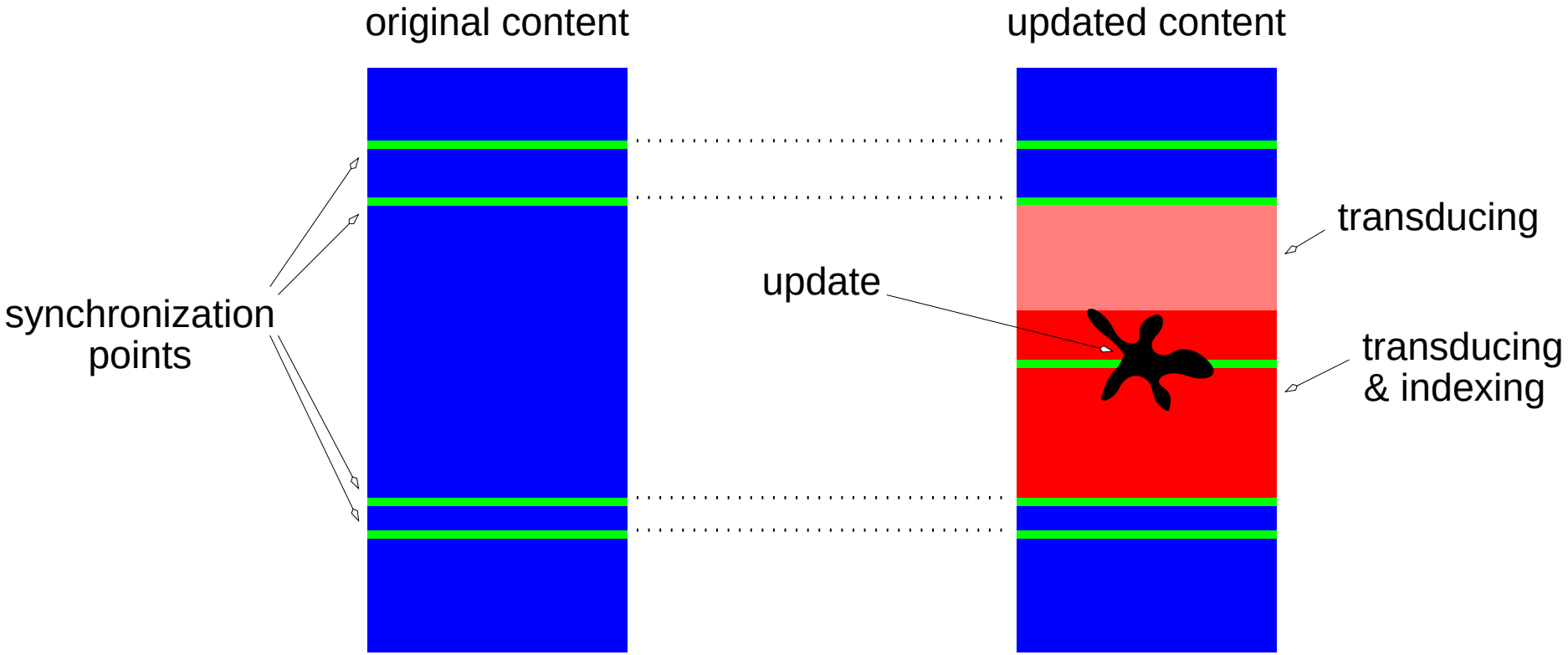
- to call transducers is costly

  define synchronization points for re-indexing

  (faster updating)

Synchronization points

places in a file where a transducer goes back to its initial state



Evaluation

- software — a user-level implementation, using PerlFS
- platform — Linux kernel 2.4, with a 2Ghz Pentium 4, 750Mb RAM, and a 40 Gb IDE disk.

Benchmark data

- a BibTeX file

  $\approx$ 8000 lines

  ```
  cd author:Jones; ls author:*
  ```

- the PofFS article

  $<$ 2000 lines

  ```
  emacs section:conclusion
  ```

- the PofFS program

  $<$ 3000 lines

  ```
  ln -s cur-poffs /poffs/!debugging/poffs.pm
  poffsmount ...
  ```

Benchmark results

- number of lines ≈ 1000 line
- number of attributes total: ≈ 1000 attr per line: ≈ 10 attr

- mount time
  from scratch $\approx 10 - 100$ s from cache $\approx 0.01 - 0.1$ s
- overhead per attribute = overhead per line ≈ 1000 byte
- `ls` response time ≈ 0.1 s
- `update` time ≈ 1 s

$\longrightarrow$ compatible with an interactive usage

negligible mount time if PofFS is used from file creation

compare with response time of `find` and `grep`

Related works

- programming language IDEs

  e.g., Smalltalk, Emacs, Eclipse

  limited query / navigation / view / updating inside files

- advanced file systems

  e.g., SFS, HAC

  limited query / navigation / updating among files

  e.g., LISFS

  advanced query / navigation / updating among files

  no view

- view update in data-bases

  easier for PofFS

Conclusions

- a running alternative to flat files

  (allows separation of cross-cutting concerns)

- a generic service

  (many types of files: programs, text, . . .

  and associated descriptions)

  system level implementation

  (applications communicate through PofFS)

- encouraging performances

- availability: http://www.irisa.fr/LIS

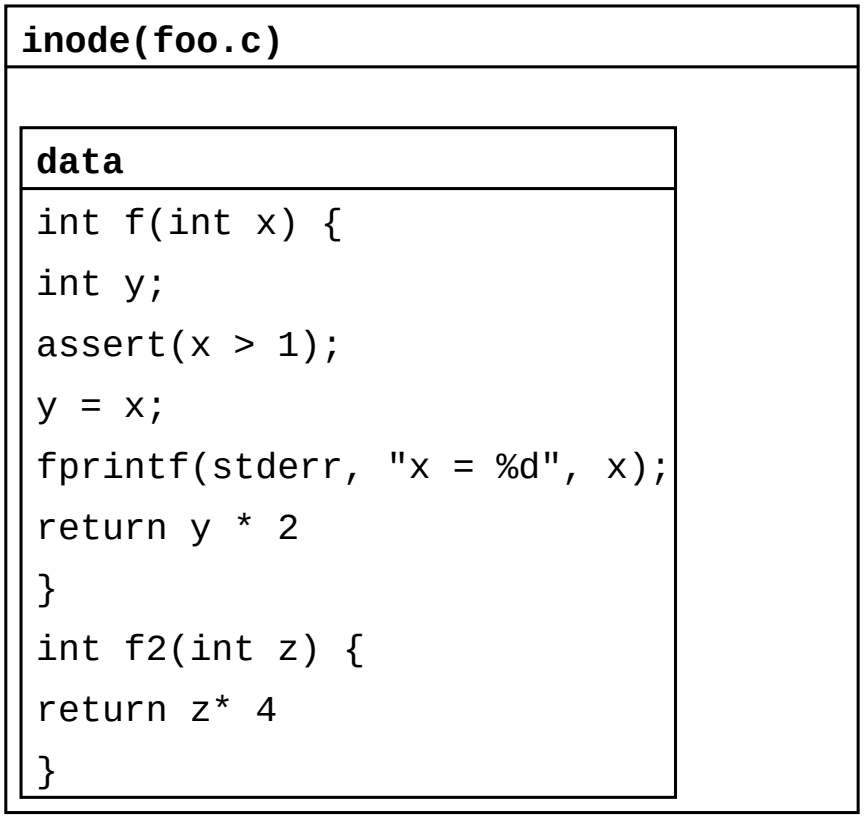
  (Logic Information Systems)

Further works

- improve performances (especially context creation)
- consider more general kinds of parts (e.g., abstract syntax node)
- explore PofFS usage for software engineering (e.g., UML, slicing, static analysis, versioning)

	BibTeX	Article	Program
Size of specific transducer	23 LoC	29 LoC	61 LoC
Size of file	269 Kbyte	78 Kbyte	92 Kbyte
Number of lines	8055	1783	2651
Mount time (1st/others)	122 s /0.216 s	22 s /0.058 s	29 s /0.078 s
Number of different attributes	7061	2132	1498
Average number of attributes per line	16	15	14
Size of file context	7800 Kbyte	1960 Kbyte	2164Kbyte
Saving time	0.956 s	0.256 s	0.331 s
`ls` response time	0.253 s	0.100 s	0.170 s
Space overhead per line	0.9 Kbyte	1.0 Kbyte	0.81 Kbyte
Space overhead	28×	25×	23×
Space overhead per attributes	1.1 Kbyte	0.9 Kbyte	1.4 Kbyte

Data-structure (1): the viewed file



Data-structure (2): the file context

	line->object
l1	o4
l2	o5
l3	o10
l4	o6
l5	o9
l6	o7
l7	o8
l8	o1
l9	o2
l10	o3

	object->contents
o1	int f2(int z) {
o2	return z * 4
o3	}
o4	int f(int x) {
o5	int y;
o6	y = x;
o7	return y * 2
o8	}
o9	fprintf(stderr, "x = %d", x);
o10	assert(x > 1);

	object->properties
o1	#function:f2, #var:z
o2	#function:f2, #var:z
o3	#function:f2
o4	#function:f, #var:x
o5	#function:f, #var:y
o6	#function:f, #var:x, #var:y
o7	#function:f, #var:y
o8	#function:f
o9	#function:f, #var:x, #debugging
o10	#function:f, #var:x, #specification

	property->objects
#function:f2	o1, o2, o3
#function:f	o4, o5, o6, o7, o8, o9, o10
#var:x	o4, o6, o9, o10
#var:y	o5, o6, o7
#var:z	o1, o2
#debugging	o9
#specification	o10

Data-structure (3): a physical view

`cd function:f/!(debugging|specification)`