Fight with Diversity
多様性と戦う

Tanaka Akira
National Institute of Advanced Industrial Science and Technology (AIST) /
Free Software Initiative of Japan (FSIJ)
RubyKaigi 2013
2013-06-01

RubyKaigi 2008 Theme

Diversity

google search

「RubyKaigi 2008 多様性　感動」

2890

By the way,

Did you have trouble with diversity of web browsers? (Ex. IE6)

"Diversity is Good" is an overstatement

Today's talk

Diversity is the Enemy

Ruby Development

- Design
  - Investigate problems
  - Find good ways to solve them
- Implementation
  - Implement on my environment
  - Porting to various environments ← This talk

dbm – a standard library

- An interface to traditional Unix database libraries
- Persistent hash table
- Key-value store

How to use dbm

- Record data in a file

```
require 'dbm'
DBM.open("dbfile", DBM::WRITER) {|d|
  d["foo"] = "bar"
}
```

- Other processes can read the data

```
require 'dbm'
DBM.open("dbfile", DBM::WRITER) {|d|
  p d["foo"]          #=> "bar"
}
```

libraries supported by dbm

Four libraries supported
One found when build is used

- ndbm: New Database Manager
- gdbm: GNU dbm
- db: Berkeley DB
- qdbm: Quick Database Manager

Ruby's directory structure

- ruby — Ruby itself
  - ext — Extension libraries
    - dbm — dbm directory
      - extconf.rb — Makefile generation script
      - dbm.c — Implementation of dbm

extconf.rb

- A script to investigate environment and generate Makefile
- Similar to configure script

Simple extconf.rb

fcntl/extconf.rb is only 2 lines

```
require 'mkmf'
create_makefile('fcntl')
```

- require mkmf library
- generate Makefile for fcntl extension library

mkmf

mkmf is a library for extconf.rb

- create_makefile generates a Makefile
- have_library tests a C library
- have_type tests a C type
- have_func tests a C function
- etc.

Top 5 of big extconf.rb

- ext/curses/extconf.rb 116 lines
- ext/openssl/extconf.rb 157
- ext/dbm/extconf.rb 254 ←
- ext/socket/extconf.rb 525
- ext/tk/extconf.rb 2057

(Ruby 2.0.0)

History of Unix dbm libraries

- 1979: Unix Version 7 provides dbm
- 1986: 4.3BSD provides ndbm
- 1990: gdbm 1.1 release
- 1991: Net/2 provides Berkeley DB
- 2003: qdbm 1.1.0 release

Unix Version 7 dbm (1979)

- AT&T license
- Only one database is usable in a process
- Database files: .pag and .dir
- Key and value size limitation:
  512 bytes for key-value pairs which keys has same hash.
  (4.2BSD expands it to 1024 bytes)
- Ruby's dbm doesn't support this library

Unix Version 7 dbm (1979)

- API: excerpt from dbm(3x)

  typedef struct { char *dptr; int dsize; } datum;
  int dbminit(file)
  datum fetch(key)
  void store(key, content)
- No header?
  (Available in source, though)
- Library: libdbm (used as -ldbm)
- struct is used as argument and return value

4.3BSD ndbm (1986)

- Derived from Unix Version 7 dbm
  AT&T license
- A process can access multiple databases
- Database files: .pag and .dir
- Standardized (Single Unix Specification)
- Ruby's dbm supports this API

4.3BSD ndbm (1986)

- API: excerpt from ndbm(3)
  #include <ndbm.h>
  DBM *dbm_open(file, flags, mode)
  void dbm_close(db)
  datum dbm_fetch(db, key)
  int dbm_store(db, key, content, flags)
- Functinos begins with dbm_
- libc contains the functions
  No linker flag required

gdbm (1990)

- GNU operating system (free Unix like OS) needs free dbm/ndbm compatible library
  (No AT&T license)
- gdbm API and dbm/ndbm compatible API
- Database files: .pag and .dir (ndbm API)
  They are hard linked
- No hardlink since gdbm 1.9
  .dir contains version information

gdbm (1990)

- Headers: gdbm.h and ndbm.h
- Library: libgdbm
- gdbm 1.8.1 splits libgdbm into libgdbm and libgdbm_compat
  libgdbm_compat contains ndbm compatible functions
- gdbm functions begins with gdbm_

Berkeley DB (1991)

- UCB has another activity to release a OS without AT&T license
  Net/2 provides Berkeley DB as replacement of ndbm
- A replacement for dbm is added later (4.4BSD Alpha, 1992)
- Rich features such as B+ tree
- Database files: .db
- Headers: db.h and ndbm.h
- libc contains the functions
- The function begins with db_

Berkeley DB 2.1 (1997)

- Released from Sleepycat Software, Inc. Sleepycat is a company formed by the authors of Berkeley DB to provide commercial support (acquired by Oracle, 2006)
- Headers: db.h only (No ndbm.h)
- Needs DB_DBM_HSEARCH definition
  #define DB_DBM_HSEARCH 1
  #include <db.h>
- Library: libdb (Not libc)

qdbm (2003)

- Rich features such as B+ tree
- Provides ndbm compatible API
- Database files: .pag and .dir
- Headers: relic.h (for ndbm API)
- Library: libqdbm

The right way to use a library

Use corresponding header and library

- include appropriate header
- link appropriate library

Difficult to be right

- ndbm.h may be ndbm, Berkeley DB 1 or gdbm
- libc may contain a ndbm compatible library
  - It may contain ndbm
  - It may contain Berkeley DB 1
  - It may not contain a ndbm compatible library
- Some legacy OS provides libndbm, which may be ndbm, Berkeley DB or gdbm

Problem example (1)

- 1997-10-29 [ruby-list:5168] 青山和光
  「RedHat の標準環境では dbm がうまくコンパイルされないようです。
  require "dbm" で、ruby: can't resolve symbol 'dbm_clearerr' となります。」
- バージョンは明示されていないが、
  One week after Ruby 1.0-971021 release

Problem example (2)

- 1999-08-16 [ruby-list:16167] 大原重樹
  「BSD/OS 3.1 上で Ruby 1.4.0 を make しようとすると、以下のように、ext/dbm の make でエラーになってしまいます。
  dbmopen.o: Definition of symbol _dbm_open (multiply defined)
  # Ruby 1.3.6 からこうなっていました。」

What's happened?

- ruby: can't resolve symbol 'dbm_clearerr'
  Berkeley DB 1's header and
  libgdbm are combined
- _dbm_open (multiply defined)
  Berkeley DB 1's header,
  libgdbm and
  Berkeley DB 1 in libc is combined

can't resolve symbol 'dbm_clearerr' ext/dbm at the time

- Ruby 1.0-971021 ext/dbm/extconf.rb

  ```
  $LDFLAGS = "-L/usr/local/lib"
  have_library("gdbm", "dbm_open") or
    have_library("dbm", "dbm_open")
  if have_func("dbm_open")
    create_makefile("dbm")
  end
  ```

- Try libgdbm first, try libdbm after that
- Don't test headers: use ndbm.h unconditionally
- Don't test dbm_clearerr
- dbm.c invokes dbm_clearerr unconditionally

can't resolve symbol 'dbm_clearerr' Declaration and definition

Declaration and definition of dbm_clearerr

- Berkeley DB 1
  - ndbm.h: No declaration ←
  - libdb: Defined in ndbm.o
- gdbm
  - ndbm.h: #define dbm_clearerr(dbf)
  - libgdbm: No definition ←
- A function in Berkeley DB
  An empty macro in gdbm

can't resolve symbol 'dbm_clearerr' The situation of the problem

- dbm_clearerr is invoked unconditionally (until Ruby 1.1b9_09)
- Berkeley DB 1 ndbm.h: No declaration for dbm_clearerr
- libgdbm: No definition for dbm_clearerr

dbm_clearerr is used but no definition

can't resolve symbol 'dbm_clearerr' Fix?

- Test dbm_clearerr at Ruby 1.1b9_10
  extconf.rb:
  ```
  have_func("dbm_clearerr")
  ```
  dbm.c:
  ```
  #ifdef HAVE_DBM_CLAERERR
        dbm_clearerr(dbm);
  #endif
  ```
  But typo
- Anyway, dbm_clearerr is not used so no link error

Afterward

- The typo is fixed at Ruby 1.3.6
- have_library("dbm", "dbm_open")
  libddbm is not a ndbm compatible library

- Not a right fix
  Both Berkeley DB and gdbm provides dbm_clearerr
  The origin of the problem is a mismatch between header and library

_dbm_open (multiply defined) Occurrence

- Ruby 1.3.6
- The typo fix for HAVE_DBM_CLAERERR causes the problem
- BSD/OS 3.1
  libc contains Berkeley DB 1 because 4.4BSD

_dbm_open (multiply defined)
Decl. and def. of dbm_open

Declaration and definition of dbm_open

- Berkeley DB 1
  - ndbm.h: DBM *dbm_open();
  - libc: Defined in ndbm.o ←
- gdbm
  - ndbm.h: DBM *dbm_open();
  - libgdbm: Defined in dbmopen.o ←

_dbm_open (multiply defined)
Decl. and def. of dbm_clearerr

Declaration and definition of dbm_clearerr

- Berkeley DB 1
  - ndbm.h: No declaration (Probably) ←
  - libc: Defined in ndbm.o ←
- gdbm
  - ndbm.h: #define dbm_clearerr(dbf)
  - libgdbm: No definition ←

_dbm_open (multiply defined) The situation of the problem

- Both dbm_open and dbm_clearerr are used
- Berkeley DB 1 header is used
  dbm_open and dbm_clearerr are functions
- Link libgdbm
- libc contains ndbm.o of Berkeley DB 1
- All functions in a object file are linked or not

_dbm_open (multiply defined) The logic of the problem

- dbm_open is used
  → dbmopen.o(libgdbm) is linked
- dbm_clearerr is used
  → ndbm.o(libc) is linked
- dbmopen.o and ndbm.o contains dbm_open
  → link error

_dbm_open (multiply defined) Fix?

- [ruby-list:16169] eban
  1. gdbm はリンクしない
  2. GDBM の ndbm.h を include する
  という解決策が考えられます. (中略)

  gdbm がリンクされるときは dbm_clearerr のテストをしないってのが簡単かな.
- [ruby-list:16170] matz
  |gdbm がリンクされるときは dbm_clearerr の
  |テストをしないってのが簡単かな.
  そうするようにします。1.4.1からは。

_dbm_open (multiply defined) Why "no test" fix the problem?

- Use Berkeley DB 1's ndbm.h and libgdbm
- Don't test dbm_clearerr because gdbm
- Don't use dbm_clearerr
- Don't link ndbm.o
- "multiply defined" is fixed

_dbm_open (multiply defined) Really no problem?

- gdbm's dbm_clearerr is an empty macro
  No invocation doesn't change the behavior
- datum structure is same
  - same fields order
  - type of dsize is both int (size_t in SUS, though)
- Constants such as DBM_INSERT are same
- Used function declarations are same

No problem

Difference in headers

- dbm_pagfno is different (but not used)
  Function to return fd for .pag file
  Not standardized but available since 4.3BSD
  - Berkeley DB 1
    ```
    #define dbm_pagfno(a)  \
      DBM_PAGFNO_NOT_AVAILABLE
    ```
  - gdbm
    ```
    extern int       dbm_pagfno ();
    ```
- The fields of DBM structure is different
  but no problem because we use only pointer

I want to use dbm_pagfno

- For setting close-on-exec flag for all fd
  - Ideal: O_CLOEXEC for dbm_open set the flag
  - Real: gdbm 1.9 ignore O_CLOEXEC
  - Workaround: Extract fd from DBM struct and set the flag
- dbm_pagfno and dbm_dirfno is required to extract fd
- Berkeley DB doesn't have dbm_pagfno
  (Because a database consists of single file)
- gdbm has both
- I am motivated to fix dbm right:
  use corresponding header and library

Situation today (1)

- gdbm (Debian)
- Berkeley DB (RedHat)
- Berkeley DB in libc (4.4BSD)
- ndbm in libc (System V)

Situation today (2)

- ndbm in libc and
  gdbm or Berkeley DB
  Many people install gdbm or Berkeley DB to avoid size limitation
- gdbm and Berkeley DB
  Recent OSes make it easy to install them via packages

Combinations
header * library * libc

- Header used
- Library explicitly linked
- ndbm compatible library in libc

Test header and library correspondence

- Test for each combination of header and library
  Link error for specification conforming code means invalid combination
  Ex: dbm_clearerr()
- Test library internals
  Ex: If a header defines _GDBM_H_, it is gdbm's header
- Avoid running for cross-compilation

Library internals vary with versions

- Headers vary with versions
- Ex: gdbm's ndbm.h defines _GDBM_H_ since gdbm 1.9.1
  _GDBM_H_ is defined by gdbm.h but it is not included from ndbm.h until gdbm 1.8.3

OS may modify headers

- Berkeley DB's ndbm.h always includes db.h and db.h defines _DB_H_
- Bug Mac OS X modifies ndbm.h not to include db.h
  So it doesn't define _DB_H_

Ignored elements

- qdbm
  It doesn't provide ndbm.h nor used in libc
- libndbm
  dbm/extconf.rb doesn't search it by default
  Several legacy OS provides libndbm, though

Kind of headers

- ndbmh ndbm
- db1h Berkeley DB 1
- db2h Berkeley DB 2 or later
- g18h gdbm 1.8.3 or former
- g19h gdbm 1.9 or later

Kind of libraries for link explicitly

- libc Don't link (use one in libc)
- db1 libdb Berkeley DB 1
- db2 libdb Berkeley DB 2 or later
- g17 libgdbm GDBM 1.8.0 or former
- g18 libgdbm GDBM 1.8.1～1.8.3 (No ndbm)
- g19 libgdbm GDBM 1.9 or later (No ndbm)
- g18c libgdbm_compat, libgdbm GDBM 1.8.1～1.8.3
- g19c libgdbm_compat, libgdbm GDBM 1.9 or later

Kind of libc

- ndbm-libc contains ndbm
- db1-libc contains Berkeley DB 1
- glibc doesn't contain ndbm compatibles (includes other libc such as uClibc)

Combinational space

- Headers 5
- Libraries 8
- libc 3
- Total 5×8×3 = 120

Table of combinational space

		libc	db1	db2	g17	g18	g19	g18c	g19c
ndbm-libc	ndbmh								
	db1h								
	db2h								
	g18h								
	g19h								
db1-libc	ndbmh								
	db1h								
	db2h								
	g18h								
	g19h								
glibc	ndbmh								
	db1h								
	db2h								
	g18h								
	g19h								

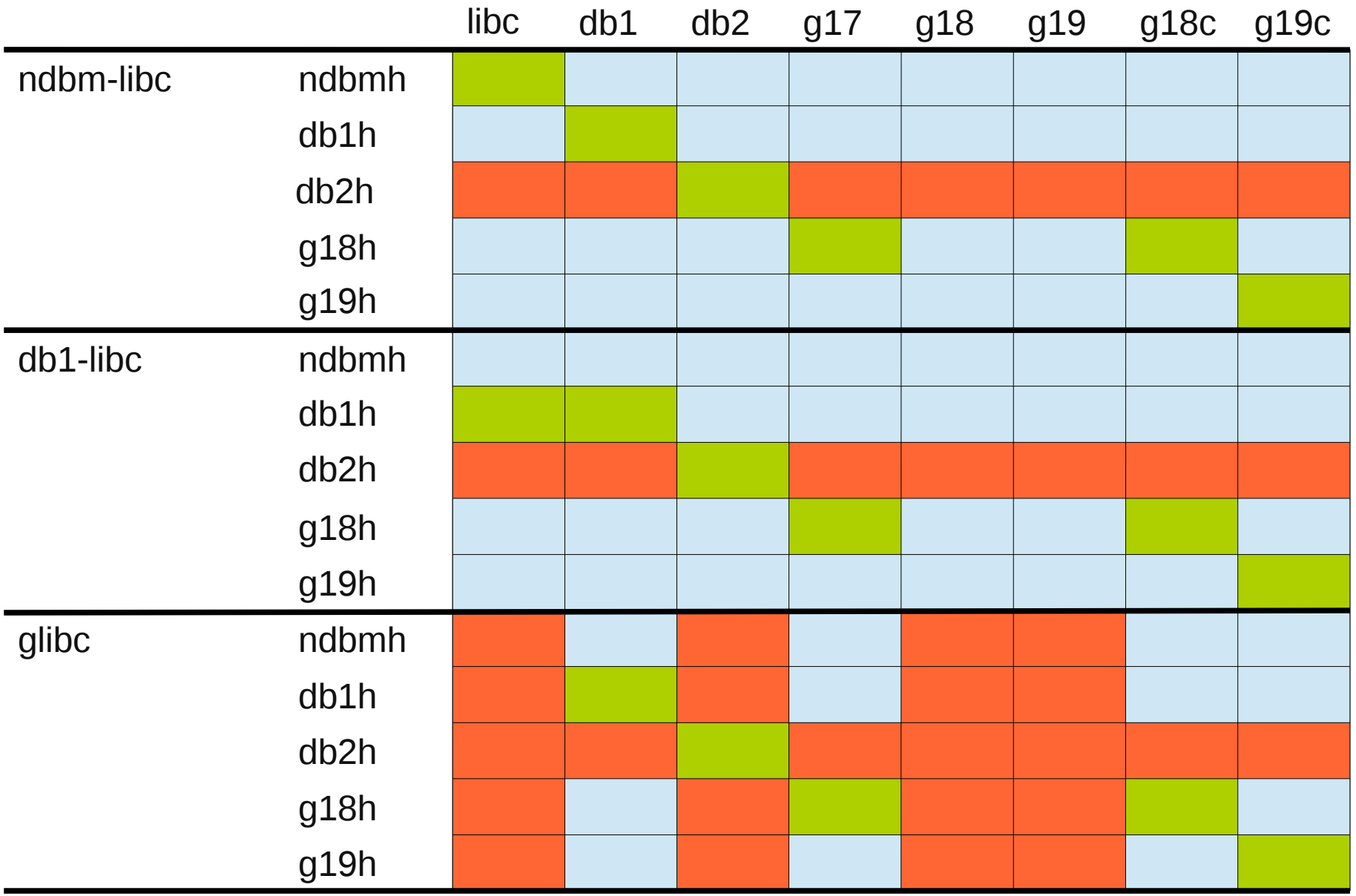
Correct Combinations

		libc	db1	db2	g17	g18	g19	g18c	g19c
ndbm-libc	ndbmh	■							
	db1h		■						
	db2h			■					
	g18h				■			■	
	g19h								■
db1-libc	ndbmh								
	db1h	■	■						
	db2h			■					
	g18h				■			■	
	g19h								■
glibc	ndbmh								
	db1h		■						
	db2h			■					
	g18h				■			■	
	g19h								■

dbm_open is linkable (1)

- Header
  ndbmh: DBM *dbm_open();
  db1h: DBM *dbm_open(const char *, int, int);
  db2h: #define dbm_open(a, b, c) \
    __db_ndbm_open(a, b, c)
  g18h: DBM *dbm_open();
  g19h: DBM *dbm_open(char *, int, int);
- db2 doesn't have dbm_open
- __db_ndbm_open is available only in db2

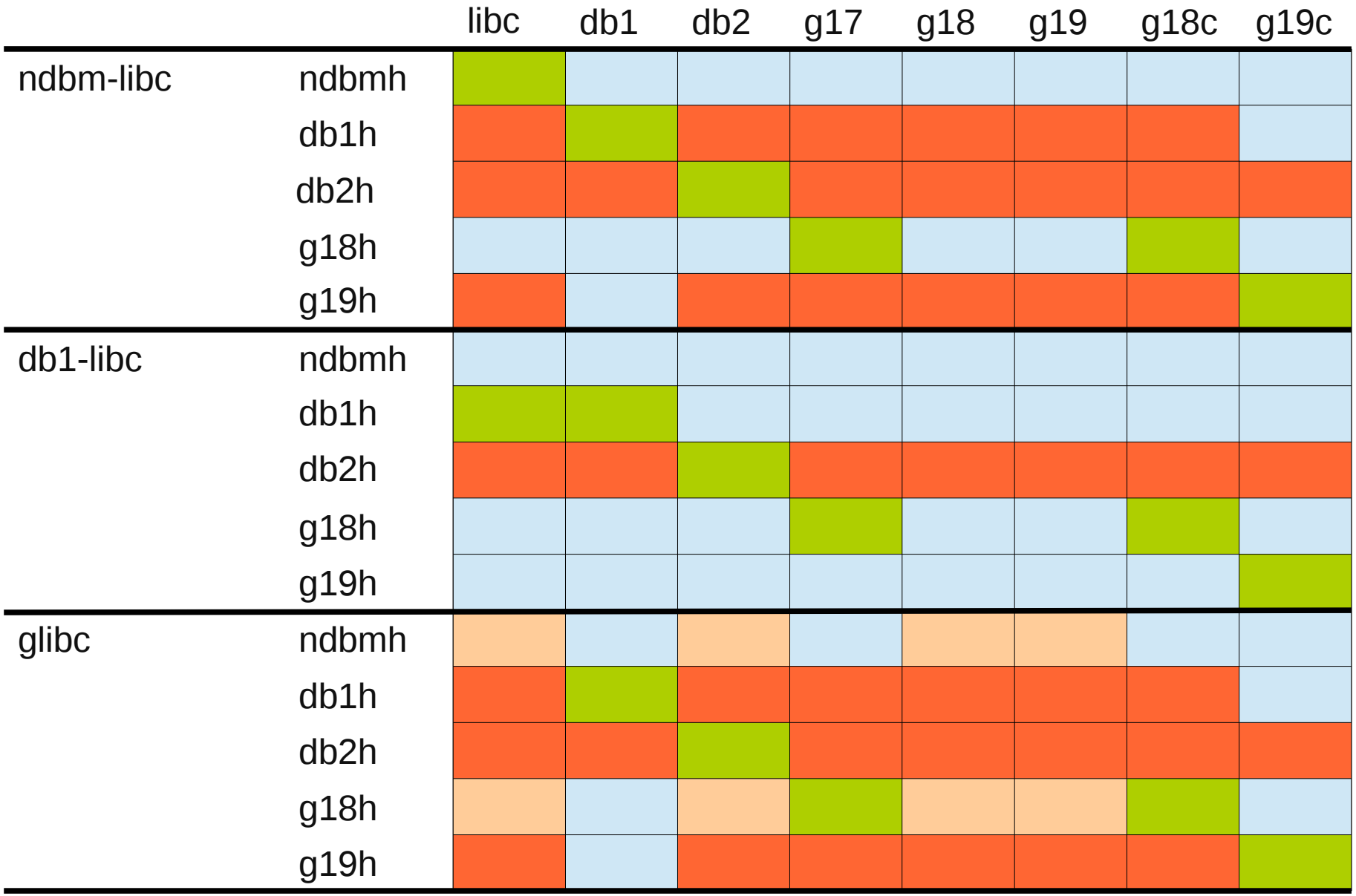
dbm_open is linkable (2)



dbm_clearerr is linkable (1)

- ndbmh: #define dbm_clearerr(db) \
  ((db)->dbm_flags &= ~_DBM_IOERR)
  db1h: Nothing
  db2h: #define dbm_clearerr(a) \
  __db_ndbm_clearerr(a)
  g18h: #define dbm_clearerr(dbf)
  g19h: void dbm_clearerr (DBM *dbf);
- ndbm, db2, g1[789], g18c doesn't have dbm_clearerr
- __db_ndbm_clearerr is available only in db2

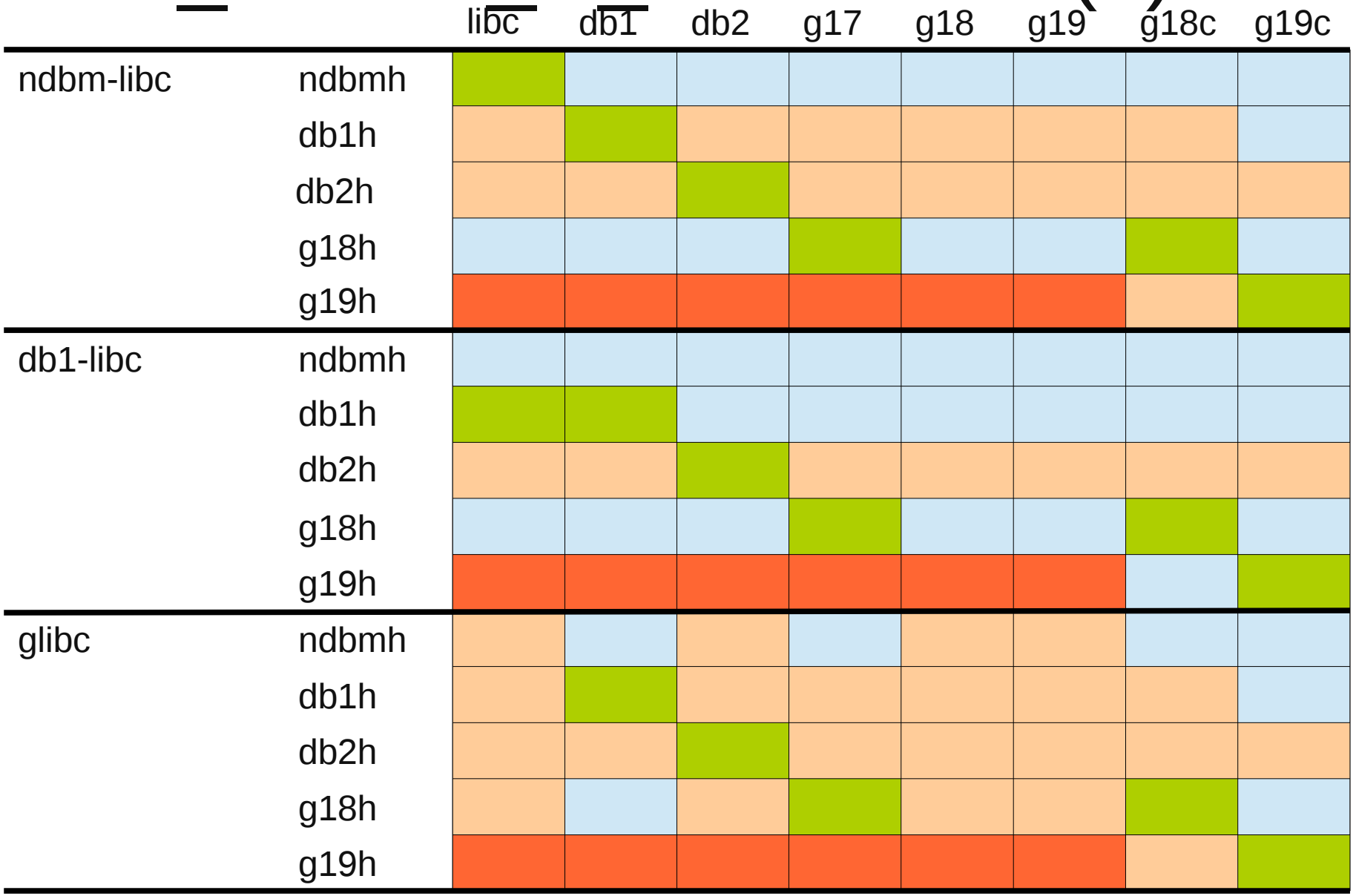
dbm_clearerr is linkable (2)



libgdbm_compat shoud be used if _GDBM_H_ is defined (1)

- Is _GDBM_H_ defined?
  ndbmh: No
  db1h: No
  db2h: No
  g18h: No (don't include gdbm.h）
  g19h: Yes (include gdbm.h)
- g1[89] doesn't have ndbm functions in libgdbm

libgdbm_compat shoud be used if _GDBM_H_ is defined (2)



gdbm_version is available for gdbm (1)

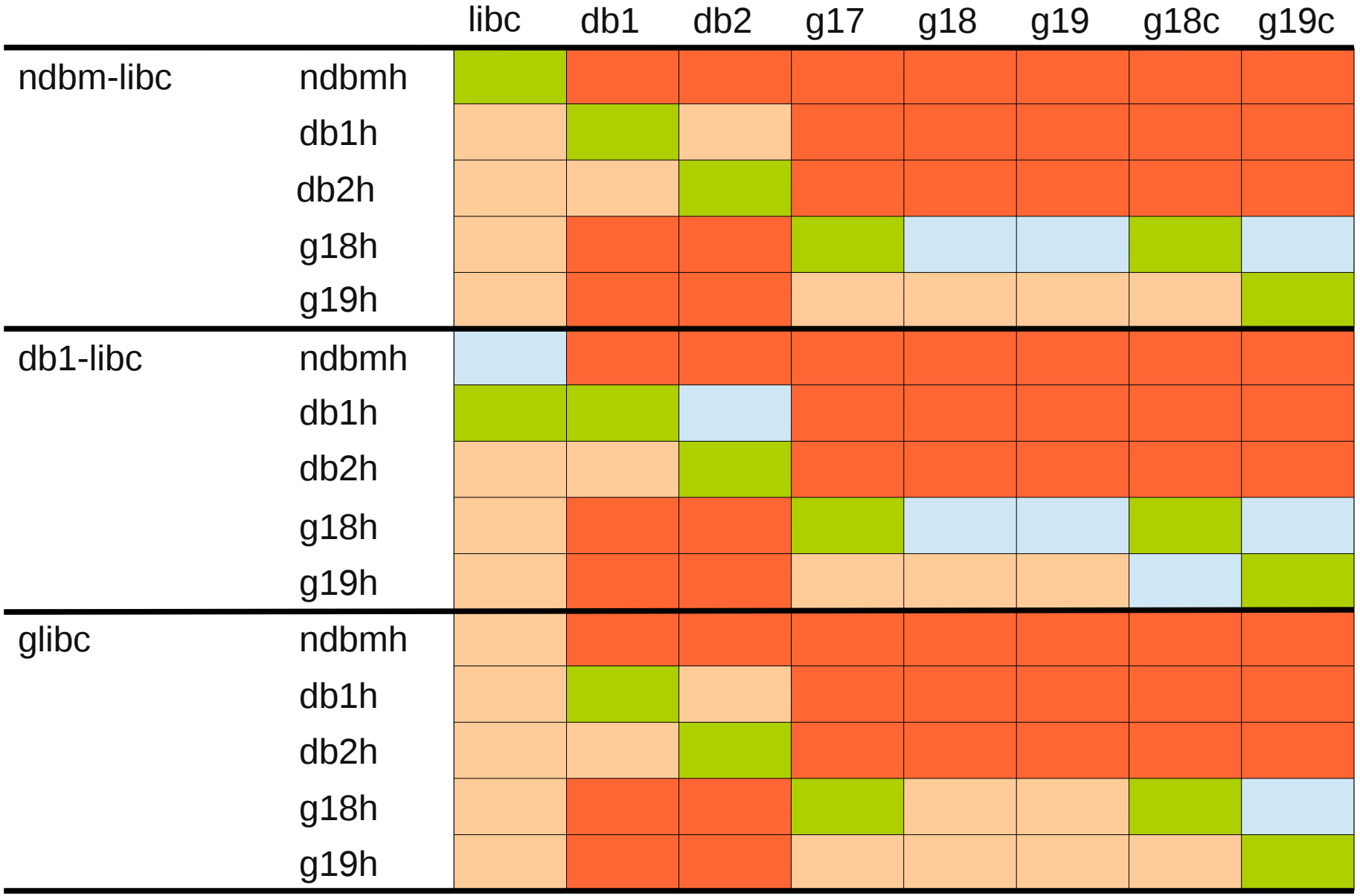
- gdbm header detection
  - g19h: _GDBM_H_
  - g18h: Is dbm_clearerr empty macro?
    Is dbm_clearerr(foobarbaz) compilable?
- libgdbm always defines gdbm_version

gdbm_version is available for gdbm (2)

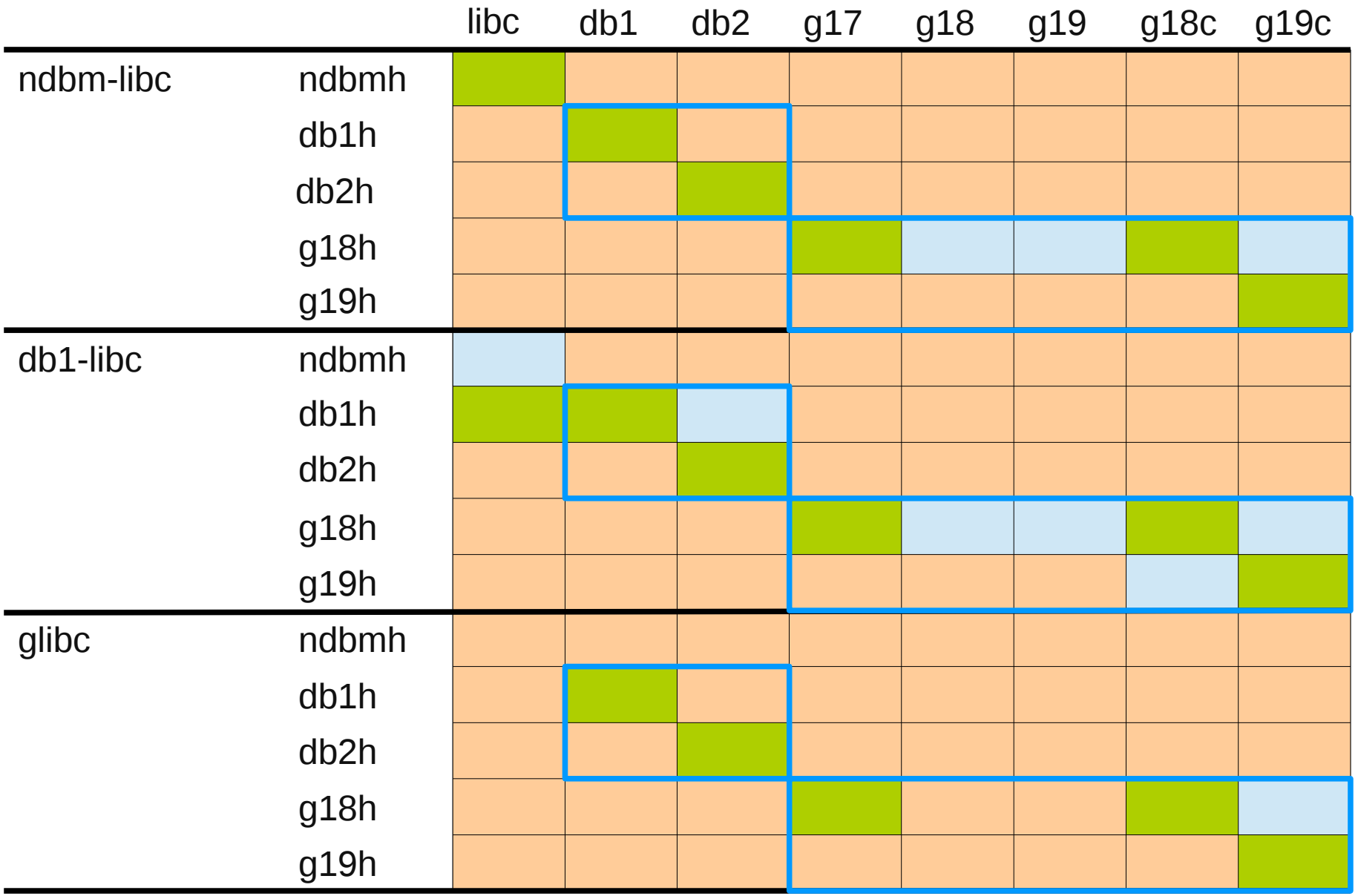
Header type and library name (1)

- Header type can be distinguished
  - g18h, g19h: as described former
  - ndbmh: Is _DBM_IOERR macro defined?
    #define _DBM_IOERR   0x2
  - db1h, db2h: Is DBM_SUFFIX macro defined?
    #define DBM_SUFFIX   ".db"
    (Don't use _DB_H_ for Mac OS X)
- libdb means Berkeley DB
  libgdbm and libgdbm_compat means gdbm

Header type and library name (2)



Current status



Current status

- extconf.rb detects many environments but is not perfect
- Environments that fail
  - Multiple versions of Berkeley DB
    Multiple versions of gdbm
    This rarely occurs because header and library are installed together
  - Install ndbm on 4.4BSD
    Combination of Berkeley DB 1 in libc and ndbm.h of ndbm

    No one installs ndbm on its own
    It is not distributed independently

Confusion of libndbm (1)

- glibc
  - 1997: glibc 2.0 provides Berkeley DB 1.85
    ndbm.h, libdb
  - 1999: glibc 2.1 provides Berkeley DB 2 too
    Berkeley DB 1: ndbm.h, libdb1
    Berkeley DB 2: db.h, libndbm is a link to libdb
  - 2000: glibc 2.2 removes them
    Because too many incompatible changes
  - Mandriva Linux provides libdb2-devel package
    and it has libndbm (for compatibility?)

Confusion of libndbm (2)

- OpenSuSE gdbm-devel package has ndbm.h and libndbm
- Tru64 UNIX provides Berkeley DB as libndbm
- SCO OpenServer, Unixware provides ndbm as libndbm

libndbm may be ndbm, gdbm or Berkeley DB if we consider legacy OS
Ruby doesn't search libndbm by default
configure --with-dbm-type=ndbm may work (not tested)

Other issues around dbm

- File formats
  No compatibility between ndbm, gdbm, Berkeley DB and qdbm
  - File extensions: pag, dir, db
  - Number of files: one or two
  - Internal format
- It may be impossible to link different libraries in a process

Link multiple libraries

- Berkeley DB dbm_open in libc is called even if --with-dbm-type=gdbm_compat is specified on FreeBSD [ruby-dev:45262]
  Summary: If multiple libraries with the same symbol are linked, only one can be referred to
- Berkeley DB 2 or later has no problem because it changes function names by macros
- gdbm splits libgdbm_compat
  No problem as long as libgdbm_compat is not used
- qdbm has a problem
  libqdbm has ndbm compatible functions

Wish for ndbm compatible libraries

- ndbm.h should define a macro to identify the library, such as XXXDBM_NDBM_H
- ndbm compatible functions should be macros as Berkeley DB 2 or later to avoid conflicts at link
- Make sure the name of the library is unique enough
  "db" is too generic

Summary

- Ruby's dbm supports several ndbm compatible libraries
- I studied various versions of the libraries
  Source code is very important
- It works well in most environments

Fight with Diversity

Diversity in many areas:

- OS increases (especially for embedded)
  - libc
  - kernel
- Many Ruby implementations

Good Luck!